NUCLEATION AND GROWTH OF AEROSOLS

Thesis by

Dale Ross Warren

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

Department of Chemical Engineering
California Institute of Technology
Pasadena, California

1986

Submitted March 31, 1986

ACKNOWLEDGMENTS

No relevant literary quote, no sports analogy here. Simply thanks to all my colleagues and friends who helped me complete this Ph. D. thesis. Special appreciation goes to the following individuals:

- my advisor, Professor John H. Seinfeld, who provided the long range guidance, motivation and general support essential to my research;
- Professor Richard C. Flagan for his unmatched insight into experimental design, wealth of ideas, and persistent good spirits;
- officemate and coworker Joe Leone, who designed so much of the experimental set-up and would find a solution for every problem that arose;
- officemate and coworker Toby Shafer, for being herself, pleasant and industrious and helpful through everything;
- to coworker Jennifer Stern, for much hard work and many good, critical suggestions.

Sincere thanks also go to (officemate?) Dr. Daniel Grosjean and to Professor Glen Cass, for providing advice, equipment, and encouragement for this project; to visiting Professor Kikuo Okuyama, for contributing his friendship, his ideas, and his experimental data for testing my model; to bygone coworkers Art Stelson, Connie Senior, and Jim Crump, for showing me the ropes; to the shop personnel, Elton Daly, Rich Eastvedt, Leonard Montenegro, and Joe Fontana, for providing an education in practical design and troubleshooting; to coworker Jeff Harrison, for being the outside expert who'd make the equipment work; and to my favorite system manager Dan Zirin, for teaching me everything I always wanted to know about the VAX (and for setting up the tennis tournaments). Appreciation for help and ideas goes to those others who went before me: Fred Gelbard, Greg McRae, Ted Russell, Peter McMurry, Pratim Biswas, Dave Strand and Mark Bassett; and to those others still present: Mark Cohen, Carol Jones, Sonya Kreidenweis, Jin-Jwang Wu, Hung Nguyen, Gidi Sageev, Yiannis Levendis, Brian Wong, Fang-dong Yin, Martha Conklin, Sue Larson, Jed Waldman, Greg Markowski, and Zoltan Szakaly. Without Lenore Kerner and Rita Mendelson in charge of communications, and Nancy Tomer and Theresa Fall taking care of my figures, I might have spent much longer here.

Never before have I worked with such exceptional people.

The National Science Foundation is gratefully acknowledged as the major sponsor of this work.

Looking back, I'd like to thank my undergraduate research advisor, Dr. Joe Katz, for getting me started on nucleation experiments. And a salute goes to two inspirational high school teachers, Richard Price and Terry Upton, who really wanted their students to think and to question.

A special acknowledgment goes to Amoco Chemicals Corporation for expediting the completion of my thesis.

And last but not least, I'd like to thank my parents, Luther A. Warren and Ruth T. Warren, who (in quite different ways) provided the values and determination that got me here; and Carol Fancher, whose friendship helped get me through.

ABSTRACT

This thesis discusses the formation of aerosol particles by homogeneous nucleation of supersaturated vapor, and the subsequent or simultaneous growth of particles by condensation. Experiments, theory, and numerical simulations are used to approach the underlying goal of understanding the aerosol evolution process in photochemically reactive systems, such as Los Angeles smog.

A comprehensive size-sectionalized model was developed for simulating the evolution of a multicomponent aerosol size distribution through homogeneous nucleation, condensational growth, coagulation, and various deposition mechanisms. When applied to atmospheric photochemistry, the model predicted that the number of new particles nucleated is controlled by the ratio between the rates of homogeneous nucleation and condensational growth. A simple model was devised for predicting the number and size evolution of particles which would be formed by a burst of homogeneous nucleation. An interesting aspect of the model was its prediction of suppression of homogeneous nucleation by seed aerosol through bulk vapor depletion. Later these predictions were verified qualitatively in two systems. One was a physiochemically well characterized system where nucleation was driven by a high initial supersaturation ratio, in which nucleation was faster than predicted by classical nucleation theory, and suppression of nucleation was only slight. The second system was our outdoor smog chamber.

In a large outdoor smog chamber, toluene and NO_x were allowed to photochemically react. Gas phase concentrations and the resulting aerosol distribution were followed with time, for various initial concentrations of reactants and seed aerosol. A few thousand seed particles per cm^3 (sub-ambient concentrations) were sufficient to suppress homogeneous nucleation that would have resulted in several times as many particles. Operation of the chamber in dual mode allowed the influence of a-single parameter, varied between the two sides of the bag, to be clearly observed, thus avoiding many of the difficulties that arise from comparing experiments conducted at different times and different temperature and sunlight histories.

TABLE OF CONTENTS

CHAPTER 1:

INTRODUCTION

An aerosol is a suspension of solid or liquid particles in a gas. Common examples of aerosols are smoke, dust and haze, to borrow from the title of the first classroom text on the subject (Friedlander, 1977). The term "aerosol" has been associated in the public mind with ecological danger since a scientific study revealed that aerosol spray cans might pose a threat to the earth's ozone layer which shields life at the planet's surface from most of the sun's carcinogenic ultraviolet radiation. (The chlorinated fluorocarbon propellants used in the spray cans caused the concern, rather than the aerosols which were delivered.) Yet aerosols themselves certainly can pose a health hazard, if they contain toxic material such as a carcinogen or radioisotope, for fine aerosols deposit themselves with a fairly good efficiency deep into the lungs, and provide a very high surface area per unit mass. Of course, aerosols also have positive uses, beginning with the hospital nebulizer which delivers medication to the throat. More significantly, aerosols formed by condensation from the vapor may allow the production of ultrapure materials (by avoiding the trace contamination that results from the handling of bulk phases) such as needed in the semiconductor and fiber optics industries.

In the Los Angeles basin in particular, aerosols have a visible impact on daily life, for it is predominantly the aerosols in smog which cause visibility degradation. Although the short term health effects of smog (shortness of breath, eye irritation, etc.) are caused predominantly by gaseous pollutants, mainly oxidants such as ozone, nitrogen dioxide, and peroxyacetyl nitrate (PAN), long term health effects may result from the deposition of particulate matter containing a wide variety of chemicals onto the lungs. While a solution of the smog problem is technologically, sociologically, and economically complex, a better scientific understanding of aerosol dynamics will facilitate a rational attack on the problem of smog.

In this work, two approaches were used simultaneously towards understanding the aerosol component of photochemical smog in particular and aerosol evolution in general. One approach was an experimental program, using an outdoor teflon smog chamber to study hydrocarbons which photooxidize in the presence of oxides of nitrogen, yielding condensable vapors and thus aerosols. Meanwhile, as preparation was being made to conduct the experiments, theoretical and numerical modeling of the system was undertaken.

Results were obtained first from the modeling approach. For the aerosol study, the major original effort was the construction of a large multicomponent computer model which approximated the continuous aerosol size distribution by an arbitrary number of fixed size sections. This model would simulate aerosol evolution by coagulation, condensational growth, homogeneous nucleation, and various deposition mechanisms, and is discussed in Chapter 2. From this big model arose an appreciation that the system behavior was dominated by the interplay between nucleation and condensation, and Chapter 3 devotes itself to this interplay and the simple model (termed the SNM model after its three dependent variables) used to describe it for a constant source of vapor. The theoretical effect of initial particles and behavior of the system in terms of dimensionless parameters, and a dimensionless vapor source/nucleation/condensation model is presented in Chapter 4.

A chance to test the predictions of the model arose with data from a system that was better characterized physically and chemically than the toluene-NO_x reaction products in the smog chamber. In this new system, a supersaturated vapor of dibutylphthalate was produced by continuous and rapid mixing. This system was amenable to simulation by the SNM model, by omitting the vapor source rate term and using initial conditions of high saturation ratios and no seed particles. These experiments and simulations are discussed in Chapter 5. Further experiments and modeling were conducted for the case where seed aerosol was initially present, and its influence on nucleation predicted and measured. This, and a detailed review of the abilities and limitations of the dimensionless model, are presented in Chapter 6.

With the physical modeling of well characterized aerosol systems completed, Chapter 7 discusses our aerosol smog chamber experiments. The simplest aromatic which produces aerosol, toluene, was selected as the initial reactant. As a companion project to this aerosol study, an experimental and theoretical investigation was conducted into the chemistry of the toluene photooxidation, and was extensively discussed in another thesis (Leone, 1984). This joint study required the equipping of a smog chamber facility with the necessary equipment to characterize the chemical composition of the gas phase and the size distribution of the aerosol phase. The key computer programs used to acquire and display and analyze laboratory data are given in Appendices A1, A2, and A3, while extensive plots of the data are provided in Appendix A4.

CHAPTER 2:

SIMULATION OF AEROSOL SIZE DISTRIBUTION EVOLUTION IN SYSTEMS WITH SIMULATANEOUS NUCLEATION, CONDENSATION, AND COAGULATION

Published in

Aerosol Science and Technology

4:31–43 (1985)

Simulation of Aerosol Size Distribution Evolution in Systems with Simultaneous Nucleation, Condensation, and Coagulation

Dale R. Warren and John H. Seinfeld
California Institute of Technology, Department of Chemical Engineering, Pasadena, CA 91125

A sectional model is presented that will simulate formation, growth, and coagulation processes for an aerosol formed by gas-to-particle conversion. Test cases have simulated a system with a source of condensable vapor, showing a burst of nucleation that is quenched by condensation onto the freshly generated and rapidly growing fine aerosol. The influence of preexisting aerosol on the size distribution evolution and on the rate of nucleation and criteria for inhibition of nucleation by initial aerosol are presented.

INTRODUCTION

An aerosol may evolve under the influence of several physical processes, such as coagulation, condensational growth, nucleation, deposition, diffusion, and convection. Aerosol dynamics are described by the so-called general dynamic equation (Gelbard and Seinfeld, 1979). Until recently, there were few cases of practical interest involving more than one or two of these processes operating simultaneously for which an evolving aerosol distribution has been computed.

In virtually all approaches to the numerical simulation of aerosol dynamics, it has been necessary to approximate the essentially continuous aerosol size distribution by a discrete spectrum. In 1980, the sectional representation for aerosol kinetics was derived as a rigorously-based algorithm for computing aerosol evolution due to coagulation (Gelbard et al., 1980). The sectional representation was notably general, allowing arbitrary specification of the size classes and of the basic aerosol property to be computed, such as number, surface area, or volume. Gelbard and Seinfeld (1980) then extended the sectional model to multicomponent aerosols, allowing, for the first time, size-composition aerosol simulations to be carried out. Implemented as a computer program (MAEROS) for following mass distribution and composition with size, it has proven to be a powerful and flexible tool for aerosol simulation (Gelbard, 1982). The MAEROS code includes treatments of coagulation due to Brownian motion, gravity, and turbulence; deposition due to settling, diffusion, and thermophoresis; condensational growth from a vapor of known supersaturation; and particle source fluxes, which can be specified to vary with time and composition. The MAEROS code does not include gas particle coupling through nucleation.

Atmospheric aerosol formation and growth have been studied in laboratory reactors, often termed smog chambers, in which gas-phase chemistry leads to the production of condensable species that may either nucleate to form new particles or condense on existing particles. Which of these two paths of gas-to-particle conversion predominates determines the nature of the resulting size distribution. Therefore, in order to understand the factors that lead to observed aerosol size distributions, it is necessary to be able to simulate aerosol size distribution evolution with simultaneous nucleation, condensation, and coagulation. The object of this work is twofold. First, we report the development of the Expanded Sectional Multicomponent Aerosol Package (ESMAP), an extension of MAEROS to include nucleation. This extension also includes a more accurate treatment of condensation than is available in the original version of MAEROS. Second, we apply ESMAP to simulate a variety of situations involving simultaneous nucleation, condensation, and coagulation. Where possible, we draw general conclusions concerning aerosol evolution in such systems.

OVERVIEW OF THE SECTIONAL MODEL

The General Dynamic Equation

An aerosol size distribution can be characterized by $n(v, \bar{x}, t)$, the number concentration at spatial coordinates $\bar{x}$ and time t of particles of volume v, where v may correspond to a discrete number of monomer units or may extend over a differential range of volume in the continuous size spectrum. The physical processes that may occur in aerosol systems are coagulation, condensation, nucleation, deposition, diffusion, and convection. The equation governing aerosol dynamics, generalized from the equation of convective diffusion, is known as the general dynamic equation (GDE) and may be written as follows (Friedlander, 1977):

$$\frac{\partial n}{\partial t} + \nabla \cdot (n\bar{v}) = \nabla \cdot D \nabla n + \left[\frac{\partial n}{\partial t}\right]_{\text{growth}} + \left[\frac{\partial n}{\partial t}\right]_{\text{coagulation}} - \left[\frac{\partial n}{\partial t}\right]_{\text{settling}}. \qquad (1)$$

If one considers an aerosol that is essentially spatially homogeneous, except perhaps near its boundaries (e.g., the container walls), the resulting differential equation contains only growth, coagulation, and deposition terms. Expressions are available for evaluating the rates of each process as a function of particle size; the problem lies primarily in reducing the essentially infinite number of interacting discrete particle sizes to a tractable model for calculations.

Sectional Representation

The sectional representation approximates the continuous aerosol size distribution with a finite number of sections. Assuming constant aerosol density ρ, it is convenient to represent the particle size with x, defined as the logarithm of the mass of the particle. Thus particle volume v is determined uniquely by x. The total mass per unit volume, present in particles of sizes no larger than x, may be defined as Q. Hence, the continuous particle mass distribution is presented in the form

$$q(x) = \frac{dQ}{dx} = \frac{dQ}{d\log(v)} = \rho\frac{d(vn(v))}{d\log(v)} = \rho v \frac{d(n(v)v)}{dv}. \qquad (2)$$

In the sectional representation, the size distribution is embodied in the mass concentration in arbitrarily selected intervals spanning the required size range. For each section l $(1 \le l \le M)$, whose size range is defined by $x_l < x < x_{l+1}$, the aerosol mass per unit volume is given by

$$Q_l = \int_{x_l}^{x_{l+1}} q(x)\, dx. \qquad (3)$$

Average rate parameters for the physical and chemical processes occurring are obtained for each section by integrating over the size range of the section. Doing so requires some assumption as to the shape of the $q(x)$ profile within a section. In the current implementation of the theory, the integration is performed assuming $q(x)$ is constant within each section, leading to the result that the sectional approximation of

$q(x)$ is

$$\bar{q}(x) = \frac{Q_l}{x_{l+1} - x_l}, \quad x_l < x < x_{l+1}. \tag{4}$$

The sectional representation reduces the infinite system of equations of the discrete or continuous GDE to a system of M equations for the sectional mass concentration vector $\bar{Q}$, having the form

$$\frac{d\bar{Q}}{dt} = \bar{f}(\bar{Q}, t). \tag{5}$$

Usually time will enter the function explicitly only if a time-dependent particle source term is present. As the number of sections is increased, the sectional representation of the GDE converges to that of the continuous equation (Gelbard et al., 1980).

The approximation of Eq. (4) defines a unique relationship between sectional mass, Q_l, and sectional number concentration, N_l, given by

$$N_l = \int_{x_l}^{x_{l+1}} \frac{q(x)}{e^x}\, dx = Q_l \frac{e^{-x_l} - e^{-x_{l+1}}}{x_{l+1} - x_l}. \tag{6}$$

Coagulation

The treatment of sectional coagulation is described in the earlier paper on multicomponent aerosol dynamics (Gelbard and Seinfeld, 1980). The coagulation terms in Eq. (5) are of the form $\beta_{ij} Q_i Q_j$, and the β_{ij} are determined from numerical evaluations of double integrals, performed a priori. The implemented form for the collision rate β uses the Fuchs–Phillips coagulation coefficient (Sitarski and Seinfeld, 1977), assuming spherical particles and unit density. In order to simplify greatly the calculation and number of the coagulation terms, a geometric constraint is imposed, requiring that each section cover a range of particle size such that the largest particle in any section will have at least twice the mass of the smallest particle included in that section, or,

$$x_{l+1} - x_l \geq \ln 2 \quad \text{for all } l. \tag{7}$$

Deposition

Given an expression for the rate of deposition (taken to include gravitational settling, diffusion to container walls or environmental surfaces, and any other relevant mechanisms) for any given particle size, assumed first-order in number concentration at the given size, a rate constant function $D(x)$ giving mass fraction lost per second for particles of size x (log particle mass) is readily formulated. The sectional deposition rate coefficient is then given by

$$\bar{D}_l = \frac{\int_{x_l}^{x_{l+1}} D(x)\, \bar{q}(x)\, dx}{\int_{x_l}^{x_{l+1}} \bar{q}(x)\, dx} = \frac{1}{x_{l+1} - x_l} \int_{x_l}^{x_{l+1}} D(x)\, dx. \tag{8}$$

Condensation and Nucleation

The simulation of particle growth requires that the vapor concentration of the condensable species be known as a function of time. Since simultaneous nucleation and condensation represent the major new features of the present model, beyond those in MAEROS, they will be discussed in some detail in the following sections.

CONDENSATION

The rate at which mass condenses onto an aerosol particle has a particle size dependence that is controlled by the Knudsen number, defined as the ratio of the condensable species mean free path, λ, to the particle radius, $\mathrm{Kn} = 2\lambda/d_p$. The condensation rate is proportional to particle surface area for a free molecule aerosol ($\mathrm{Kn} \gg 1$), but proportional only to particle diameter for a continuum regime aerosol ($\mathrm{Kn} \ll 1$). For general aerosol modeling, an expression that applies throughout the free molecule, transition, and continuum regimes is required. and the Fuchs–Sutugin interpolation formula (Fuchs and Sutugin, 1971) has been adapted for use here. For constant density particles of diameter d_p and corresponding mass m_p, a growth constant H is used to express the fractional rate of aerosol mass increase,

$$\frac{dm_p}{dt} = Hm_p. \tag{9}$$

By the Fuchs–Sutugin formula,

$$H = \frac{12D_1[p_1 - p_d]f_0(\mathrm{Kn})m_1}{d_p^2\rho_l kT} \tag{10}$$

$$f_0(\mathrm{Kn}) = \frac{1 + \mathrm{Kn}}{1 + 1.71\,\mathrm{Kn} + 1.33\,\mathrm{Kn}^2}. \tag{11}$$

Here D_1 is the monomer diffusivity in air (or the predominant gas species), $[p_1 - p_d]$ is the difference between the partial pressure and that at the particle surface for the condensable species, ρ_l is the monomer density as a liquid (or solid), and m_1 is the molecular mass of the monomer.

Since the Fuchs–Sutugin formula was derived from simple kinetic theory for self-diffusion, proper convergence to the two limiting cases of a free molecule aerosol (Kn $\rightarrow \infty$) and of a continuum aerosol (Kn $\rightarrow 0$) requires that the following condition apply: $\lambda = 3D_1/\overline{c_1}$, where $\overline{c_1}$ is the mean speed of a monomer molecule, equal to $(8kT/\pi m_1)^{0.5}$. Although this λ generally differs from the monomer mean free path from rigorous kinetic theory, it allows the use of the simple interpolation formula of Eq. (11), which, as shown by Pesthy et al. (1983), gives mass fluxes similar to more rigorous transition regime expressions, which are discussed by Davis (1983).

To evaluate condensation coefficients based on Eq. (10), the excess partial pressure $[p_1 - p_d]$ must be known. Thus, in order to compute aerosol condensation coefficients a priori, we assume a fixed S_{ref}, a reference saturation ratio, and thus set $(p_1 - p_d) = (S_{\mathrm{ref}} - 1)p_0$ as the reference excess pressure, where p_0 is the saturation vapor pressure. At any time, the actual aerosol condensation coefficient (for now neglecting the Kelvin effect) will be the stored sectional condensation coefficient multiplied by $(S - 1)/(S_{\mathrm{ref}} - 1)$, where S is the actual system saturation ratio at the given time.

The sectional coefficient for intrasectional condensational growth is given by $\overline{H}_l$ for section l, evaluated as

$$\overline{H}_l = \frac{\int_{x_l}^{x_{l+1}} q(x)H(x)\,dx}{\int_{x_l}^{x_{l+1}} q(x)\,dx}. \tag{12}$$

Assuming, as usual, that $q(x)$ may be approximated as $Q_l/(x_{l+1} - x_l)$, then

$$\overline{H}_l = \frac{1}{x_{l+1} - x_l}\int_{x_l}^{x_{l+1}} H(x)\,dx. \tag{13}$$

Hence $\overline{H}_l Q_l$ is the mass rate at which the condensable species (at saturation ratio S_{ref}) grows onto particles in section l. However, some of the largest particles in section l will grow into section $l + 1$ because of condensation. This intersectional condensation rate, $\bar{I}_{l+1}$, is given simply by

$$\bar{I}_{l+1} = Hq, \quad \text{evaluated at } x = x_{l+1}. \tag{14}$$

The overall mass balance for section l with respect to condensation, hence, is

$$\left[\frac{\partial Q_l}{\partial t}\right]_{\mathrm{condensation}} = \overline{H}_l Q_l - \bar{I}_{l+1} + \bar{I}_l. \tag{15}$$

Note, Q_l represents the total mass within a size section, summed over all components. For particle growth processes, the multicomponent aspect of the model needs mention. The $\overline{H}_l Q_l$ term is composed exclusively of the condensable species, growing onto particles of any composition; the intersectional $\bar{I}$ term (like coagulation and deposition terms) is nonspecific, composed of all chemical species in direct proportion to their concentration, evaluated at the boundary between section l and $l + 1$. Note that the first section receives only the freshly nucleated particles, consisting of the condensable species. Particles that would grow out of the largest section are usually retained in that section to achieve mass conservation, making $I_{M-1} = 0$; this is an example of finite domain error (Gelbard and Seinfeld, 1978), and is insignificant, provided that almost no mass grows into the largest section.

The evaluation of the intersectional condensation rate $\bar{I}$ is a critical decision for a sectional model. The process of condensational growth is analogous to the advection process, and presents a notoriously difficult problem to solve numerically, because of the tendency for numerical diffusion and dispersion to arise. A sectional model, which handles the coagulation process effectively, is not currently able to simulate the condensation process with great accuracy in a reasonable number of sections.

At the boundary between sections, the assumed sectional mass concentration profile, $\bar{q}(x)$, undergoes a discontinuity. Assuming $q(x_{l-1}) = \bar{q}_l$ (first-order approximation) leads to a significant amount of numerical diffusion, with the size distribution becoming broader than it should be. For cases where a small number of sections is used, this first-order representation even allows uninterrupted condensation to cause an unlimited increase of mass in a small section, rather than the eventual depletion that must occur as all the particles grow out of the section. Clearly, the true concentration profile at the boundary between two sections must lie between the mean concentrations of the two sections (excluding cases where the sectional model is used with an insufficient number of sections to provide adequate size resolution of the aerosol distribution). The solution arrived at in MAEROS was to use a linear interpolation (or second-order approximation) for $\bar{q}$ which is described by

$$\bar{q} = f\bar{q}_l + (1-f)\bar{q}_{l-1}, \quad \text{at } x_{l+1} \tag{16}$$

where

$$f = \frac{x_{l-2} - x_{l+1}}{x_{l-2} - x_l} \tag{17}$$

$$\bar{I}_{l+1} = \frac{fQ_l + (1-f)Q_{l-1}}{x_{l+1} - x_l} H(x_l). \tag{18}$$

Here $0 < f < 1$, and $f = 0.5$ for the usual geometrically similar set of sections. (Note that the first-order approximation uses $f = 1$.) This approximation is quite adequate for slow rates of condensation, where condensation is not causing the smaller particles to disappear entirely by growth to much larger sizes. However, for rapid condensational growth, this second-order model proves to be numerically unacceptable. Inevitably, as a smaller (upstream) particle size section is depleted by condensational growth, its mass eventually becomes negligible compared to that in the next section. Yet, the second-order condensation model bases the intersectional condensation rate on the average of the upstream and the downstream sections. The relatively larger mass in the downstream section (which can receive mass but not supply it) will try to drive the upstream mass concentration negative. Attempts to assure nonnegativity such as employing minimum time constants for depletion only offer slight relief; the second-order method is intrinsically vulnerable to numerical dispersion, and only partially alleviates numerical diffusion. (The dispersion is manifested as the trailing edge of an aerosol peak artificially breaks up into a series of lesser peaks.)

Since conservation of number is a very important property for an aerosol dominated by condensational growth, this condition may be used to set the rate of intersectional condensation, while retaining the intrasectional condensation rate integral of Eq. (13). Summed over all sections, the apparent gains in particle number due to the intrasectional condensation mass terms, $\bar{H}_l Q_l$, must exactly cancel the apparent net losses in particle number associated with the intersectional condensation mass terms, $\bar{I}_{l+1}$. The relationship between sectional number concentration and sectional mass concentration is given by Eq. (6), which allows one to define a number mean particle mass, $\bar{m}_l$, for section l, by the following:

$$\bar{m}_l = \frac{Q_l}{N_l} = \frac{x_{l+1} - x_l}{e^{-x_l} - e^{-x_{l+1}}}. \tag{19}$$

Converting Eq. (15) to a particle number balance by Eq. (19) gives, for section l,

$$\left[\frac{dN_l}{dt}\right]_{\text{condensation}} = \frac{\bar{H}_l Q_l}{\bar{m}_l} + \frac{\bar{I}_l}{\bar{m}_l} - \frac{\bar{I}_{l-1}}{\bar{m}_l}. \tag{20}$$

Total number concentration is conserved by the condensation process. Thus summing Eq. (20) over all sections will yield a zero rate of change in total number. If we assume that $\bar{I}_{l-1}$ is a linear function of $\vec{Q}$, a generalization of Eq. (18), it becomes apparent that $\bar{I}_{l-1}$ can only depend on Q_l, since when Q_l goes to zero, $\bar{I}_{l+1}$ must go to zero lest Q_l assume negative values, as nonexistent mass grows into the downstream section. Since each element of $\vec{Q}$ is linearly independent, the sum of all terms containing Q_l must equal zero for each l when Eq. (20) is summed over all sections, and thus,

$$\frac{\bar{H}_l Q_l}{\bar{m}_l} + \frac{\bar{I}_{l+1}}{\bar{m}_{l+1}} - \frac{\bar{I}_{l-1}}{\bar{m}_l} = 0. \tag{21}$$

Hence, the intersectional flux is given by

$$\bar{I}_{l+1} = \frac{\bar{H}_l Q_l}{1 - e^{-(x_{l-2} - x_l)/2}}. \tag{22}$$

This will be called the number-conserving expression for condensation. It assures particle number conservation while maintaining the mass balance (with an unaffected rate of gas-to-particle mass conversion) that is primary to the sectional aerosol models. Its main drawback is that, as a simple first-order sectional expression, it allows numerical diffusion to spread out the aerosol distribution to some degree, which can be restrained by using a larger number of aerosol sections. (If the geometric constraint is maintained to simplify the number and calculation of coagulation coefficients, it may not be possible to increase the number of sections.)

The new comprehensive sectional model reported here allows inclusion of the Kelvin effect, which reduces the condensational growth rate of small particles. Since the extent of the Kelvin effect depends on the saturation ratio S in a nonlinear manner, it cannot be included in the a priori sectional integrals for the condensation rate coefficients. This Kelvin factor is given by

$$\frac{p_1 - p_d}{p_1 - p_0} = \frac{S - e^{d_c/d_p \ln S}}{S - 1} = \frac{S - S^{d_c/d_p}}{S - 1}. \tag{23}$$

Here d_c is the critical diameter calculated from homogeneous nucleation theory, and d_p is a mean diameter for the section. (Since the diameters of successive sections are usually set at a constant ratio, a geometric mean diameter for the section seems a sensible choice for d_p, and is used in ESMAP.) If the Kelvin factor is less than zero, indicating evaporation of aerosol, an alternate expression is used for the intersectional flux. This number-conserving evaporation expression is derived in an analogous manner from Eq. (20) on the argument that an evaporative $\bar{I}_{l+1}$ can depend only on Q_{l+1}. (See Eq. (25).)

In addition to being able to represent rapid condensational growth, ESMAP computes the vapor concentration of the condensable species by solving its differential conservation equation, assuming a generation rate for the condensable species is provided. In the MAEROS model, a changing vapor concentration can be handled only by repeatedly interrupting the time integration to adjust S_{ref} to S and rescale the condensation coefficients. ESMAP automatically scales the condensation coefficient at each time step, using S calculated internally from the mass concentration of the vapor. Hence the actual implementation of condensation now uses the MAEROS form for the intrasectional condensation coefficients (Eq. 13) scaled by a factor z_l given by

$$z_l = \frac{S - S^{d_c/d_p}}{S_{\text{ref}} - 1}. \tag{24}$$

In the case that $d_p < d_c$, evaporation can occur if section l contains any of the condensable species. (If there is nothing to evaporate, z_l is set to zero.) Assuming that the condensable species forms a volatile shell on each particle in the size section, and assuming a negative $\bar{I}_{l+1}$ must depend on Q_{l+1} rather than Q_l, the number-conserving expression for evaporation becomes

$$\bar{I}_l = \frac{\bar{H}_l Q_l}{1 - e^{(x_{l-1} - x_{l-1})/2}}. \qquad (25)$$

For either condensation or evaporation, the intersectional $\bar{I}_{l-1}$ fluxes are computed from the intrasectional $\bar{H}_l$ or $\bar{H}_{l-1}$ fluxes after the latter have been scaled by z_l.

The total mass condensation rate, R'_c, at any time is given by

$$R'_c = \sum_{l=1}^{M} z_l \bar{H}_l Q_l. \qquad (26)$$

NUCLEATION

Several alternate theoretical expressions have been used for the estimation of homogeneous nucleation rates. (See Springer (1978) for a review of classical and revised homogeneous nucleation theories.) For the purposes of this paper, the classical nucleation expression will suffice.

$$J = ZP_{g_c} = ZB_{g_c} n_1 e^{-W'_c}. \qquad (27)$$

J denotes the rate of nucleation (cm^{-3} sec^{-1}), g_c is the critical cluster number, and n_1 is the monomer concentration. The frequency, B_g, at which a cluster containing g monomers collides with a monomer, is given by simple kinetic theory as

$$B_g = \beta_1 s_g = \frac{n_1 \bar{c}_1}{4} s_1 g^{2/3}. \qquad (28)$$

The critical cluster number, g_c, is given by

$$g_c = \left[\frac{2\Theta/3}{(\ln S)}\right]^3. \qquad (29)$$

The dimensionless surface energy Θ, proportional to the surface tension σ, times the monomer surface area s_1 (extrapolated from the liquid density) is defined as

$$\Theta = \frac{\sigma s_1}{kT}. \qquad (30)$$

The dimensionless energy barrier to nucleation W'_c, decreasing strongly with rising saturation ratio S, is given by

$$W'_c = \frac{4\Theta^3}{27(\ln S)^2}. \qquad (31)$$

The Zeldovich nonequilibrium factor is given by

$$Z = \left(\frac{W'_c}{3\pi}\right)^{0.5} / g_c. \qquad (32)$$

Homogeneous nucleation is integrated into the sectional model as a particle source term into the smallest sectional size. The nucleation term is treated as an $\bar{I}_0$ term, a flux of particles into the smallest size section. Like intersectional condensation terms, consistency in particle number concentration is a fundamental consideration. The mass flux due to homogeneous nucleation is set so as to make the apparent number flux (based on the average particle size in the smallest section) equal to the number of particles nucleated by Eq. (27). It is necessary that the minimum sectional diameter used be somewhat larger than the largest critical diameter for nucleation which may result in a significant number of particles, lest the freshly nucleated particles fail to continue to grow. This formulation assumes that the cluster populations near the critical size are in a steady state distribution determined by the monomer concentration. This steady state assumption is implicit in the classical treatment of homogeneous nucleation. The generalized sectional model extends the assumption of a steady state distribution past the critical size up to the minimum sectional size used, so some care should be taken in the selection of the minimum sectional diameter, especially if the nucleation is only borderline steady state. An approximate criterion for steady state nucleation is Warren and Seinfeld (1984),

$$\tilde{R}_s = \frac{R_s}{n_{\text{sat}}^2 \bar{c}_1 s_1/4} < 1. \qquad (33)$$

R_s is the volume generation rate of condensable molecules, and $\tilde{R}_s$ is called the dimensionless source rate, since it represents the ratio of the source rate to a characteristic collision rate. n_{sat} is the saturated number concentration of the condensable species. The source rate R_s also allows us to define the characteristic time that it takes for the source to replenish the saturation concentration,

$$\tau_s = \frac{n_{\text{sat}}}{R_s}. \tag{34}$$

SIMULATION OF AEROSOL DYNAMICS DURING GAS-TO-PARTICLE CONVERSION

Comparison with a Monodisperse Model

A series of simulations was conducted with the comprehensive sectional model to follow the gas-to-particle conversion process with time, and to follow the size distribution of the resulting aerosol. In an earlier paper (Warren and Seinfeld, 1984), a simpler condensation plus nucleation model which assumed a monodisperse aerosol was used to estimate the number and size of particles formed by a constant source rate with no initial aerosol present. The simple monodisperse model follows total mass and total number, and treats the aerosol as if it were all at one size. The sectional codes not only allow the aerosol to have a spectrum of sizes, but they even artificially widen the size range. The sectional expressions for condensation are subject to noticeable numerical diffusion. The numerical diffusion goes with the square root of the number of sections traversed, so quadrupling the number of sections that span a size range will halve the numerical diffusion.

The sectional aerosol code consistently predicted higher resulting number concentrations by a factor of two or three for the many-orders-of-magnitude range of number concentrations which were simulated. This discrepancy was attributed to the different degrees of polydispersion associated with the aerosol models. For given total number and total mass concentrations, it is the monodisperse distribution that maximizes both the surface area and the diameter-number product, and thus maximizes the condensation rate. Since condensation and nucleation are competing for the available condensable vapor, a monodisperse distribution will cause homogeneous nucleation to be quenched faster by more rapid condensational growth, and result in a lower total number concentration. Thus the monodisperse model should underpredict the number of particles nucleated, while a sectional model will tend to overpredict the number of particles formed.

Test Cases for Sectional Condensation

Simulating the growth of a monodisperse distribution is a difficult but informative test for a sectional code, which is not intended for following sharp wavefronts. To allow an analytic solution for comparison, the following special assumptions are made: (1) no coagulation, (2) no nucleation, (3) no Kelvin effect, (4) a fixed saturation ratio, and (5) free molecule aerosol growth laws apply. Under these assumptions, the diameter of each particle grows linearly with time. Figure 1 shows the calculated sectional size distributions for the case of a monodisperse aerosol which is given time to grow from an initial diameter of 0.01 μm to a diameter of 1.01 μm, corresponding to just over a 10^6 increase in mass. The exact analytic size distribution is a delta function, indicated by arrows at the proper initial and final diameters. The curves correspond to a sectional representation using 9, 18, and 36 sections per decade of diameter size, starting with the same initial mass and number concentrations. The mass distribution has been normalized so resulting mass (area under the curves) should be unity. It should be noted that total number was conserved exactly. Figure 2 shows the increase in the number mean particle diameter with time for the three different sectional resolutions and for the analytic solution. The simulated growth rate corresponds to a 10^{-2} μm sec^{-1} increase in particle diameter, so each particle grows by its initial diameter each second. Figure 3 shows the deviation in aerosol mass from the analytical solution as a function of time for the same simulations. After a million-fold increase in mass, which corresponds to Figure

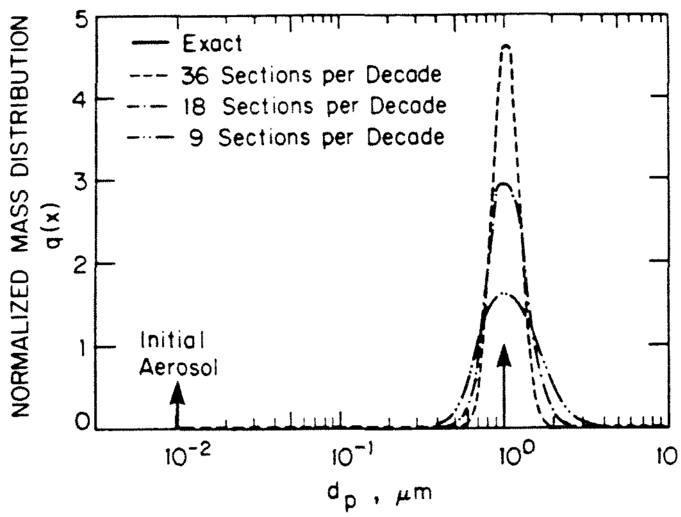


FIGURE 1. Condensational growth test showing normalized sectional mass distributions for a monodisperse aerosol whose diameter has increased one-hundred-fold.

1, the sectional aerosol mass is 0.656, 0.821, and 0.909 of the analytic value, for the resolutions of 9, 18, and 36 sections per diameter decade, respectively. The mean diameter goes as the cube root of the mass concentration, so the mean diameter predictions are quite good—a 3% error after growing one-hundred-fold in diameter for the high resolution case. Thus we see the rate of increase in total mass, and hence apparent mean size, is slightly underpredicted by the sectional model, as expected, since the broadening of the size distribution caused by sectionalization must reduce the predicted overall rate of condensation. These effects may be lessened by going to a larger number of sections. The condensation results presented here represent a substantial improvement over those obtained with Eq. (18) as used in MAEROS.

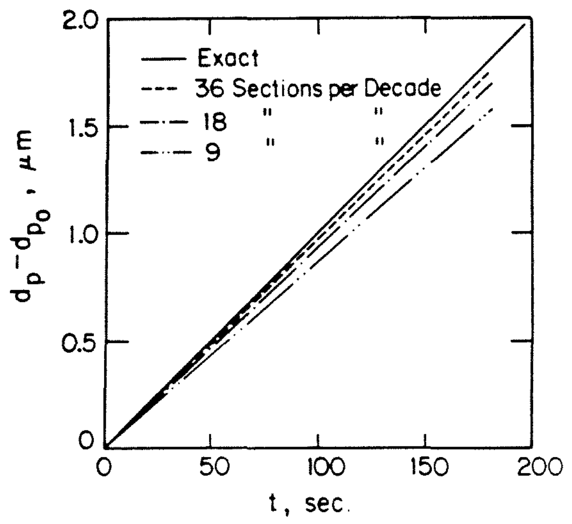


FIGURE 2. Aerosol diameter increase with time for the condensational growth test.

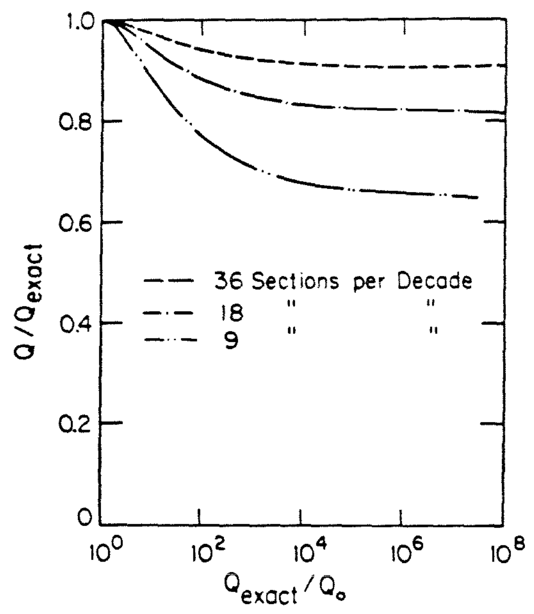


FIGURE 3. Deviation of sectional aerosol mass from the analytic solution with time for the condensational growth test.

Nucleation of a Vapor in the Absence of Preexisting Aerosol

We will now consider a very simple but important case where homogeneous nucleation can occur. It consists of a simple batch reactor having a constant rate of generation of condensable material and no initial aerosol, a situation approximating many smog chamber experiments. For simplicity, a fairly typical model organic compound will be assumed, having the properties listed in Table 1. Only the rate of its generation (from gas phase chemistry) will be varied in these simulations. It is assumed that condensation and nucleation are the only important processes. In practical cases, deposition could be very important, but its inclusion does not add to our qualitative understanding of the system. Our simulations have consistently shown that coagulation is entirely negligible for a system undergoing steady state nucleation.

Figures 4 and 5 show the aerosol mass distributions at several times for dimensionless source rates of 0.1 and 0.01, respectively. The dimensionless time is scaled to the time that it takes for the source to regenerate the saturation concentration, τ_s. While this scaling causes the total aerosol mass to be similar at a given dimensionless time, the mean particle sizes and the total number conentrations which result will vary widely. For $\tilde{R}_s = 0.1$, about 5×10^4 cm^{-3} particles are produced within 10 minutes. For $\tilde{R}_s = 0.01$, about 700 cm^{-3} particles nucleate within 1 hour. For an even slower source rate, corresponding to $\tilde{R}_s = 0.001$ (not shown), about 10 cm^{-3} large particles are generated over an 8-hour span. Lower dimensionless source rates give each nucleated particle more time to grow, and since larger particles remove vapor more quickly, they result in lower peak supersaturations and substantially less total nucleation.

TABLE 1. Physical Properties of the Model Compound

Property	Symbol	Value	Units
Temperature	T	298	K
Total pressure	p	1	atm
Molecular weight	M_1	100	g gmole^{-1}
Liquid density	ρ_l	1	g cm^{-3}
Diffusivity	D_1	0.0760	cm^2 sec^{-1}
Surface tension	σ	25	dyne cm^{-1}
Vapor pressure	p_0	0.0001	dyne cm^{-2}
Dimensionless surface energy	Θ	8.878	
Characteristic collision time	τ_β	4.48	seconds

Nucleation of a Vapor in the Presence of Preexisting Aerosol

A sufficient quantity of a preexisting aerosol will prevent homogeneous nucleation by depleting the vapor phase, so the saturation ratio will never exceed unity by enough to allow homogeneous nucleation to occur. On the other hand, a very small amount of aerosol will not influence the system significantly. The interesting situations lie between these two extremes. Intuitively, it would seem that an aerosol number concentration somewhat less than that resulting from homoge-

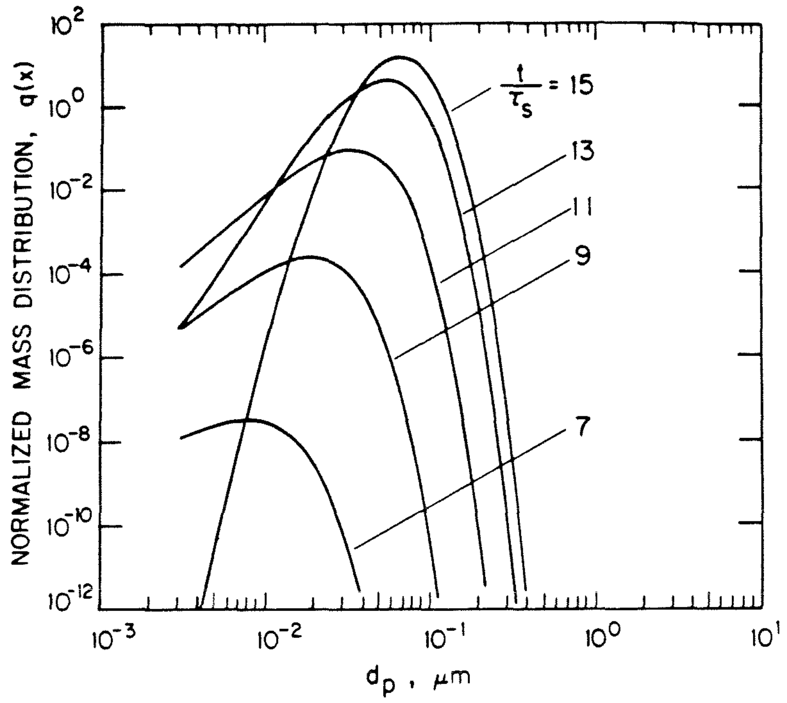
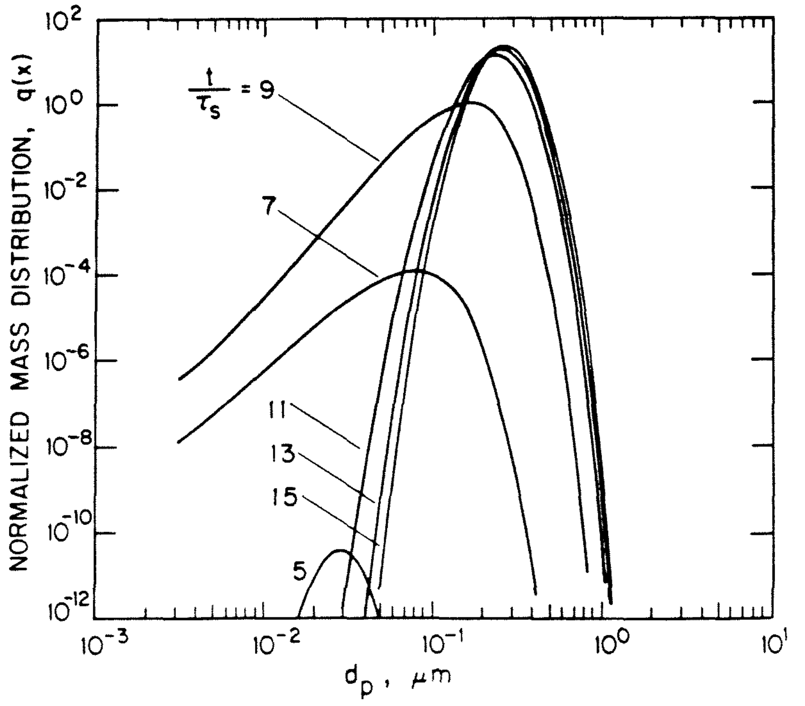


FIGURE 4. Evolution of the normalized aerosol mass distribution by nucleation and condensation in a particle free system with $\tilde{R}_s = 0.1$ using 36 sections per decade.

FIGURE 5. Evolution of the normalized aerosol mass distribution by nucleation and condensation in a particle free system with $\tilde{R}_s = 0.01$ using 36 sections per decade.



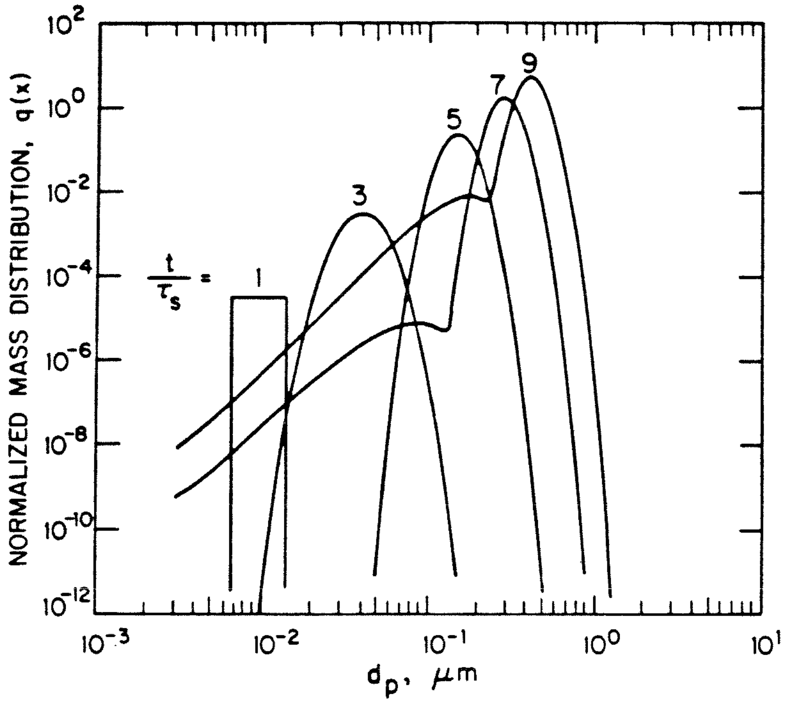
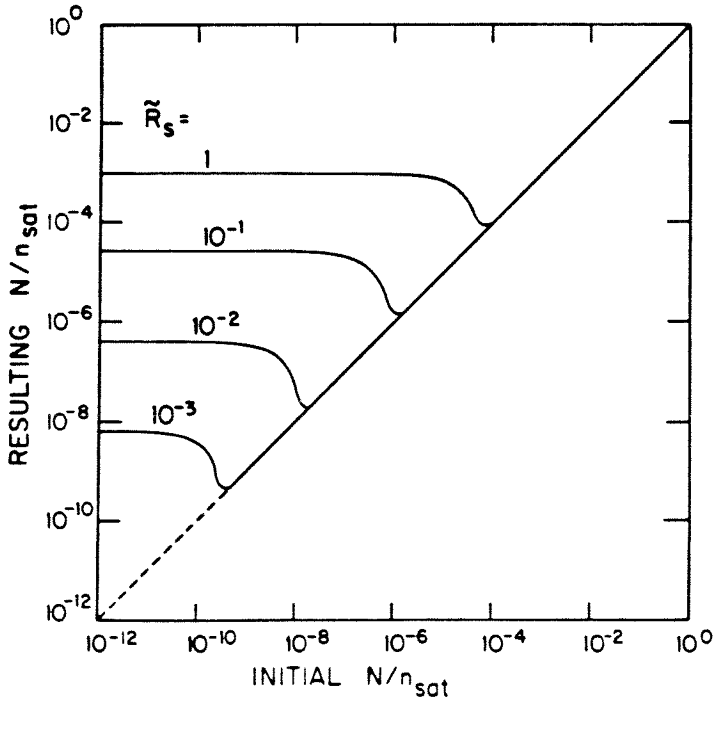


FIGURE 6. Evolution of the normalized aerosol mass distribution by nucleation and condensation in a system with initial aerosol and $\tilde{R}_s = 0.01$ using 36 sections per decade.

FIGURE 7. Resulting dimensionless aerosol number concentration as a function of initial dimensionless number concentration for various $\tilde{R}_s$.



nous nucleation in the particle free system should greatly inhibit homogeneous nucleation. This turns out to be correct. Figure 6 illustrates this case with an initial aerosol loading of only 10^{-5} μg m^{-3} of 0.01 μm aerosol, for a initial number concentration of 28 cm^{-3}, or 4% of the resulting number for a particle free system with the same 0.01 dimensionless source rate. The total resulting number concentration becomes 40 cm^{-3}, for better than an order of magnitude reduction in particle number (but essentially the same particulate mass loading). Figure 7 shows resulting total number conentrations after the burst of nucleation (if any) versus initial number concentrations for a variety of dimensionless source rates. The resulting total number concentrations pass through a minimum that is roughly an order of magnitude lower than the particle free case, and for which little nucleation occurs. Any preexisting particles will grow fairly large by the time the saturation ratio can rise high enough to allow nucleation. Fewer of these large particles are necessary to result in a total rate of condensation greater than the generation rate, thus quenching nucleation with fewer particles than would otherwise result.

CONCLUSIONS

The sectional multicomponent aerosol model has been significantly expanded by algorithms for homogeneous nucleation and condensation which are coupled to the vapor phase and which accurately conserve particle number concentration. This new ESMAP model allows one to examine the balance between new particle formation and existing particle growth, as well as coagulation and deposition mechanisms, as needed. For a nucleating system, the resulting number concentration and size distribution can be predicted as a function of the source rate and initial aerosol concentration. Interestingly, the resulting number concentration should go through a minimum for an initial number concentration somewhat less than what would result in the absence of a preexisting aerosol.

This work was supported by National Science Foundation grant ATM-8208625.

REFERENCES

Davis, E. J. (1983). Transport phenomena with single aerosol particles. *Aerosol Sci. Technol.* 2:121–144.

Friedlander, S. K. (1977). *Smoke Dust and Haze.* Wiley-Interscience, New York.

Fuchs, N. A., and Sutugin, A. G. (1971). High-dispersed aerosols. In *Topics in Current Aerosol Research.* G. M. Hidy and J. R. Brock, eds., Pergamon, Oxford, vol. 2, p. 34.

Gelbard, F., and Seinfeld, J. H. (1978). Numerical solution of the dynamic equation for particulate systems. *J. Comp. Phys.* 28:357–375.

Gelbard, F., and Seinfeld, J. H. (1979). The general dynamic equation for aerosols. *J. Colloid Interface Sci.* 68:363–382.

Gelbard, F., Tambour, Y., and Seinfeld, J. H. (1980). Sectional representations for simulating aerosol dynamics. *J. Colloid Interface Sci.* 76:541–556.

Gelbard, F., and Seinfeld, J. H. (1980). Simulation of multicomponent aerosol dynamics. *J. Colloid Interface Sci.* 78:485–501.

Gelbard, F. (1982). *MAEROS User Manual.* Sandia National Laboratories, SAND80-0822, Albuquerque, NM.

Pesthy, A. J., Flagan, R. C., and Seinfeld, J. H. (1983). Theory of aerosol formation and growth in laminar flow. *J. Colloid Interface Sci.* 91:525–545.

Sitarski, M. and Seinfeld, J. H. (1977). Brownian coagulation in the transition regime. *J. Colloid Interface Sci.* 61:261–271.

Springer, G. S. (1978). Homogeneous nucleation. In *Advances in Heat Transfer*, T. F. Irvine and J. P. Hartnett, eds., Academic Press, New York, vol. 14, p. 281.

Warren, D. R., and Seinfeld, J. H. (1984). Nucleation and growth of aerosol from a continuously reinforced vapor. *Aerosol Sci. Technol.* 3:135–153.

Received 30 April 1984; accepted 21 May 1984

CHAPTER 3:

NUCLEATION AND GROWTH OF AEROSOL FROM A CONTINUOUSLY REINFORCED VAPOR

Published in

Aerosol Science and Technology

3:135–153 (1984)

Nucleation and Growth of Aerosol From a Continuously Reinforced Vapor

Dale R. Warren and John H. Seinfeld
California Institute of Technology, Department of Chemical Engineering, Pasadena, CA 91125

In this paper the dynamic coupling of the vapor and aerosol phases in spatially uniform systems undergoing new particle formation and driven by a constant source of vapor is considered. In such a system, freshly nucleated particles are sites of rapid condensation, causing vapor depletion and the cessation of homogeneous nucleation. Modifications of classical nucleation theory to account for cluster scavenging by the aerosol are evaluated. Extensive numerical experiments show that cluster scavenging by aerosols generally has negligible effect on the resulting aerosol distribution and that, except for the very earliest stages of nucleation, condensation dominates over nucleation as the route of gas-to-particle formation even in initially particle-free systems. The total number and size of the resulting particles are shown to depend strongly on the source rate, vapor pressure, and surface tension.

INTRODUCTION

A gas-to-liquid phase transition may occur when a vapor becomes supersaturated, with a partial pressure exceeding the equilibrium vapor pressure. If the supersaturation is small, the phase change will occur only in the presence of a sufficient quantity of "foreign" condensation nuclei, such as aerosol particles, charged ions, or extended surfaces; this is known as heterogeneous nucleation. Homogeneous nucleation, or self-nucleation, whereby clusters of the vapor itself serve as nucleation sites, occurs only with appreciable supersaturation. Homogeneous nucleation has a large activation energy arising from the Kelvin effect, which is the enhancement of the evaporation rate of small clusters because of curvature.

Nucleation may occur in many types of systems. The condensable material may be generated only within a brief period of time, as in shock tubes and some combustion processes. These may be called pulse reactors. In other systems, such as diffusion cloud chambers and smog reactors, condensable material is continuously transferred to or generated within the nucleation zone. These may be called continously reinforced reactors. The main subject of this paper is the simplest type of continously reinforced reactor, the constant-rate aerosol reactor, as described by Friedlander (1982). The constant-rate aerosol reactor is assumed to have constant temperature, constant pressure, and constant source rate and is spatially uniform in composition. The smog reactors used in many aerosol studies often approximate constant-rate aerosol reactors.

In a constant-rate aerosol reactor, the initial homogeneous nucleation, if it occurs at all, will have a limited duration. The accumulation and growth of freshly nucleated particles will introduce sufficient particle surface area, so that further particle growth by condensation will deplete the vapor concentration below the threshold level for homogeneous nucleation. A basic description of the behavior of such a system should include the total number of particles nucleated, the time interval during which the nucleation occurs, and the average resultant particle size.

In this paper we describe the early stage of the evolution of an aerosol by the competing processes of new particle formation and particle growth as it occurs within a constant-rate aerosol reactor. A simple dynamic model for the coupled vapor-cluster-aerosol system is developed and compared with a model employing a more rigorous size resolution of the aerosol. The extension of classical homogeneous nucleation theory to include cluster scavenging by the aerosol is discussed. The system is analyzed in terms of nondimensional parameters, in which form general predictions are presented for the size, concentration, and rate at which aerosol is produced in such a system.

CONDENSATION

Condensation will be defined here as the growth of existing particles (of any composition) by the addition of the condensable vapor, which will also be referred to as the monomer. The dividing line between condensation and nucleation depends on the size at which a growing cluster is considered to become a new particle, so further growth will be onto an "existing particle." This size must be at least as large as the critical size (which varies), but preferably no larger than the smallest detectable particle size. It is convenient to define the minimum particle size as a fixed diameter d_s slightly above any critical nucleus diameter for which significant homogeneous nucleation occurs.

Condensational growth, expressed as a rate of increase in mass per particle, is proportional to particle surface area in the free molecule regime, but proportional to particle diameter in the continuum regime. An expression that applies to the free molecule, transition, and continuum regimes is required, and the Fuchs–Sutugin interpolation formula (Fuchs and Sutugin, 1971) has been adapted for use here. For a particle of diameter d_p, the net flow per second of monomer to the surface, F, is approximately given by the Fuchs–Sutugin formula,

$$F = 2\pi D_1 d_p \frac{(p_1 - p_d)}{kT} f_0(Kn) \qquad (1)$$

$$f_0(Kn) = \frac{1 + Kn}{1 + 1.71Kn + 1.33Kn^2}. \qquad (2)$$

Here D_1 is the monomer diffusivity in air (or the predominant gas species), $[p_1 - p_d]$ is the difference between the particle pressure and that at the particle surface for the condensable species, and Kn is the Knudsen number given by $2\lambda/d_p$. In order for the expression for F to attain the correct free molecule limit as $Kn \to \infty$, the mean free path λ is defined by

$$\lambda = \frac{3D_1}{\overline{c_1}} \qquad (3)$$

where $\overline{c_1}$ is the monomer mean kinetic velocity. The mean free path defined by Eq. (3) is equal to the mean free path of monomer molecules by simple kinetic theory for self-diffusion. Nevertheless, we shall employ Eq. (3), an approach similar to that of Pesthy et al. (1983).

By kinetic theory, the net flow of monomer to a particle is given by

$$F_{\text{kin}} = \frac{\overline{c_1}}{4} \frac{[p_1 - p_d]}{kT} \pi d_p^2. \qquad (4)$$

The Fuchs–Sutugin formula is readily shown to reduce to the following correction factor to the above condensation rate obtained in the kinetic limit:

$$f(Kn) = \frac{F}{F_{\text{kin}}} = \frac{1.333Kn + 1.333Kn^2}{1 + 1.71Kn + 1.333Kn^2}. \qquad (5)$$

For each particle size, F varies with time for a constant-rate aerosol reactor only as a result of the pressure difference term in Eq. (1). As will be discussed in the next section, this term may be evaluated as

$$[p_1 - p_d] = p_0[S - S^{d_c/d_p}]. \qquad (6)$$

The saturation ratio, $S = p_1/p_0$, is simply the monomer partial pressure divided by the equilibrium (planar) vapor pressure p_0 of the monomer. d_c is the critical diameter for homogeneous nucleation. For particles much larger than the critical size,

$$[p_1 - p_d] = p_0[S-1], \qquad d_p \gg d_c. \tag{7}$$

The mass rate of condensation onto a particle of mass m_p is simply expressed by

$$\frac{dm_p}{dt} = m_1 F. \tag{8}$$

The total mass rate of condensation R'_c onto aerosol in a system is found by an integral of the mass growth rate of each particle over the aerosol number density distribution dN/dd_p,

$$R'_c = m_1 \int_{d_s}^{\infty} F(d_p) \frac{dN}{dd_p}\, dd_p. \tag{9}$$

R'_c can be expressed as the difference between two terms, R'_{c-} proportional to p_1, which corresponds to the collision rate, and R'_e proportional to p_d, which corresponds to the reverse process of evaporation,

$$R'_{c+} = \frac{m_1 \bar{c}_1 p_0 S}{4kT} A_E \tag{10}$$

$$A_E = \int_{d_s}^{\infty} f(Kn)\pi d_p^2 \frac{dN}{dd_p}\, dd_p. \tag{11}$$

The kinetic-equivalent total surface area is denoted by A_E, which will equal the total aerosol surface area A for a free molecule aerosol. If the Kelvin effect may be ignored, Eq. (7) permits the evaluation of R'_e and R'_c to be

$$R'_e = \frac{m_1 \bar{c}_1 p_0}{4kT} A_E \tag{12}$$

$$R'_c = \frac{m_1 \bar{c}_1 p_0}{4kT} A_E [S-1]. \tag{13}$$

NUCLEATION

A key aspect of the work presented here is the dynamic modeling of the nucleation process for an evolving aerosol, including the modification of nucleation rate due to the scavenging of clusters. For this reason, it is useful to review certain elements of nucleation theory in this section.

Classical Homogeneous Nucleation Theory

The classical theory of homogeneous nucleation developed by Volmer, Becker, Doring, and Zeldovich around the 1930s remains the most common approach for treating homogeneous nucleation. Classical nucleation theory begins by considering a system of monomers and clusters in a supersaturated system. A true equilibrium cannot exist here, as the supersaturated state is inherently unstable with respect to phase change. However, a hypothetical, pseudoequilibrium state is supposed, constrained such that no clusters are allowed to exceed a certain size. The concept of pseudoequilibrium allows thermodynamics to be brought to bear on nucleation, an essentially nonequilibrium process.

Classical nucleation theory determines the pseudoequilibrium number concentration, n_g^e, of clusters containing g monomers ("g-mers"), from the reversible work W_g (or equivalently the free energy change) required to form a liquid g-mer from the supersaturated vapor. Cluster concentrations are assumed to follow a Boltzmann distribution. We note that in classical theory W_g is simply the change in the sum of bulk free energy plus surface energy terms,

$$\begin{aligned} W_g &= g(\mu_{\text{liq}} - \mu_{\text{gas}}) + \sigma s_g \\ &= -gkT \ln S + \sigma s_1 g^{2/3}. \end{aligned} \tag{14}$$

The chemical potential is denoted by μ, the cluster surface tension by σ, and the g-mer surface area by s_g. These cluster properties are assumed to be the same as the bulk thermodynamic properties, and s_1 is a monomer surface area, extrapolated from the bulk liquid mean volume per monomer v_1 and assuming a spherical shape. Nondimensionally,

$$\frac{W_g}{kT} = -g \ln S + g^{2/3}\Theta \tag{15}$$

where

$$\Theta = \frac{\sigma s_1}{kT} \tag{16}$$

$$n_g^e = n_1^e e^{-W_g/kT}. \tag{17}$$

Thus g-mer formation has a favorable phase-transition energy term proportional to the cluster number times logarithm of the saturation ratio (S) and an unfavorable surface energy term proportional to cluster number to the two-thirds power times a characteristic surface energy number Θ (surface energy corresponding to the extrapolated monomer surface area s_1 in kT units). Once the free energy barrier to nucleation, arising from surface energy, is overcome, and a cluster exceeds the critical size, further growth is increasingly favored. The critical cluster, existing at the peak of the activation energy curve for nucleation, has a cluster number and corresponding (dimensionless) activation energy given by

$$g_c = \left[\frac{2\Theta/3}{(\ln S)}\right]^3 \tag{18}$$

$$\tilde{W}_c = \frac{W_{g_c}}{kT} = \frac{(2\Theta/3)^3}{2(\ln S)^2} = \frac{4\Theta^3}{27(\ln S)^2}. \tag{19}$$

Classical nucleation theory assumes that two major dynamic processes are occurring, which will be in balance at pseudoequilibrium. They are the collisional growth of a cluster by incorporating a monomer, and the evaporative shrinking of a cluster by loss of a monomer. These forward and reverse pseudoequilibrium cluster growth fluxes are equal and denoted by P_g, in units of number/volume/time, where g is the number of monomers in the cluster prior to monomer addition or after evaporation. Cluster–cluster collisions and simultaneous evaporation of more than one monomer from a cluster are neglected. The pseudoequilibrium situation may be depicted as

$$\cdots n^e_{g-1} \underset{P_{g-1}}{\longleftrightarrow} n^e_g \underset{P_g}{\longleftrightarrow} n^e_{g+1} \cdots.$$

While n^e_g may be calculated by Eq. (17), P_g is given by

$$P_g = B_g n^e_g. \tag{20}$$

B_g is the frequency of monomer addition for a g-mer, obtained from kinetic collision theory and will be discussed with the kinetic derivation for classical nucleation. The monomer concentration n^e_1 is virtually equal to the total concentration of the condensable species n (neglecting a few percent existing as dimer and larger clusters), which is proportional to the saturation ratio.

At pseudoequilibrium, there is no net cluster growth, since the forward and reverse rates P_g cancel each other out for any g. Classical nucleation theory proceeds to calculate a steady rate of nucleation J from the pseudoequilibrium distribution, assuming that once a cluster exceeds the critical size it will tend to continue to grow. Clusters smaller than the critical size are assumed to approach their pseudoequilibrium values, whereas clusters larger than the critical size cannot build up to an appreciable concentration (which would be large at pseudoequilibrium) because they rapidly grow larger. Thus

$$J \approx P_{g_c}. \tag{21}$$

A more accurate value for J is obtained by including the Zeldovich factor Z in the nucleation expression to account for the gradual manner in which steady state cluster concentrations n depart from pseudoequilibrium cluster concentrations n^e with increasing g, rather than assume the sudden jump in concentrations from pseudoequilibrium at g_c to zero at $g_c + 1$, implied by Eq. (21). The Zeldovich factor is given by

$$Z = \left(\frac{\tilde{W}_c}{3\pi}\right)^{0.5} \Big/ g_c = \left(\frac{\sigma}{kT}\right)^{0.5} \frac{2v_1}{\pi d_c^2}. \tag{22}$$

Typically the Zeldovich factor is of order 0.01 to 0.1 for systems undergoing nucleation. The rate of nucleation from the classical theory is thus given by what we shall call the standard classical nucleation rate expression,

$$J = ZP_{g_c} = ZB_{g_c} n_1 e^{-\tilde{W}_c}. \tag{23}$$

Conceptually, the steady state rate of nucleation depends on the pseudoequilibrium concentration of critical clusters times the frequency at which each critical cluster collides with a monomer times the Zeldovich nonequilibrium correction factor.

The Kinetic Derivation of Classical Nucleation

Classical nucleation theory is also readily derived using a kinetic population balance for the steady state concentrations n_g of the clusters, each consisting of g monomers. (This form of derivation was used in the original Becker–Doring–Zeldovich classical nucleation model. The standard closed form of Eq. (23) can be obtained through the use of some additional simplifying approximations, which shall not be made in this section.) The cluster situation may be depicted as follows:

$$\cdots \xrightarrow[I_{g-1}]{} n_{g-1} \xrightarrow[I_g]{} n_g \xrightarrow[I_{g+1}]{} n_{g+1} \xrightarrow[I_{g+2}]{} \cdots$$

$$\frac{dn_g}{dt} = I_g - I_{g+1} = 0 \qquad \text{at steady state.} \tag{24}$$

The droplet current I_g is defined as the net rate of growth of clusters into size g from size $g-1$ and is found as the difference between the rates of monomer addition to a $(g-1)$-mer and evaporation from a g-mer,

$$I_g = B_{g-1}n_{g-1} - E_g n_g. \tag{25}$$

E_g is the frequency of evaporation from a g-mer. The monomer addition frequency for a g-mer, B_g, is calculated from kinetic theory as the rate constant for the collision of monomer with g-mer. An approximate expression (obtained from hard sphere kinetic collision theory, assuming cluster diameters much less than the mean free path, which applies for atmospheric pressures and below), used by Carlton (1980), is

$$B_g = \zeta_c n_1 \frac{\overline{c_1}}{4} s_1 \left(1 + \frac{1}{g}\right)^{0.5} (1 + g^{1/3})^2 \tag{26}$$

$$\overline{c_1} = \left(\frac{8kT}{\pi m_1}\right)^{0.5}. \tag{27}$$

Conceptually, the collision rate is proportional to an accomodation coefficient ζ_c times the monomer concentration, n_1 times a mean kinetic velocity $\overline{c_1}$ multiplied by the cluster surface area $s_g = s_1 g^{2/3}$ times an enhancement factor. The enhancement factor, classically neglected, arises as the monomer is treated as a small sphere instead of a point, and the cluster is allowed a small thermal motion instead of being treated as a fixed surface. The accomodation coefficient for monomer collisions is unity if all collisions are effective. For $g \gg 1$ (for particles that are, nevertheless, much smaller than the mean free path), the enhancement factor falls to near unity as the g-mer diffusivity and the monomer radius and surface area become comparatively insignificant and assuming an accomodation coefficient of 1, the formula simplifies to the commonly used

$$B_g = \beta_1 s_g = \beta_1 s_1 g^{2/3} \tag{28}$$

where

$$\beta_1 = \frac{n_1 \overline{c_1}}{4}. \tag{29}$$

The standard classical nucleation treatment resulting in Eq. (23) uses the two above equations, where β_1 is the frequency of monomer collisions per surface area.

The evaporation frequency E_g is related to B_g by including the Kelvin effect (a two-term Taylor expansion of the Gibbs–Thompson relationship for vapor pressure lowering above a curve surface),

$$E_g = B_g \frac{p_d}{p_1} = B_g S^{(g_c/g)^{1/3} - 1}. \tag{30}$$

For an interface with no curvature, $E_g = B_g$ is necessary for equilibrium to exist, which also requires $S = 1$. Since the evaporation rate constant for an interface should be independent of S, while the monomer addition rate constant is proportional to S, E_g/B_g observes the proper $1/S$ dependence, neglecting surface curvature.

The steady state distribution of g-mer concentrations n_g described by equations (24) and (25) defines a tridiagonal system of linear algebraic equations. Closure of the set of equations is obtained using $n_1 = n$ and assuming $n_G/n_1^e \to 0$ for some $G \gg g_c$. A solution may be obtained by direct numerical means, or by calculating the pseudoequilibrium state as an intermediate procedure. The latter method will be outlined here because of its similarity to the thermodynamic derivation of classical nucleation theory.

The distribution of g-mer concentrations under pseudoequilibrium conditions, denoted by the use of n_g^e rather than n_g, obeys the following:

$$I_g = 0, \qquad \frac{dn_g^e}{dt} = 0. \tag{31}$$

This leads directly to

$$\frac{n_g^e}{n_{g-1}^e} = \frac{E_{g-1}}{B_g}. \tag{32}$$

Since $n_1^e = n$, the n_g^e distribution is defined. Returning to the steady state nucleation case, let

$$U_g = \frac{n_g}{n_g^e}. \tag{33}$$

Recalling Eq. (25),

$$I_g = B_{g-1}n_{g-1}^e U_{g-1} - E_g n_g^e U_g. \tag{34}$$

Hence,

$$\begin{aligned} I_g &= B_{g-1}n_{g-1}^e\left[U_{g-1} - U_g\right] \\ &= P_{g-1}\left[U_{g-1} - U_g\right]. \end{aligned} \tag{35}$$

Note $U_1 = 1$, since the monomer concentration is a known input to either the n_g or n_g^e distribution. If g is sufficiently large, say at $g = G$, there will be a negligible concentration of g-mers. This $U_G = 0$ is the Szilard boundary condition, leading to closure of the set of population balance equations. If the actual system is at steady state, meaning a constant droplet current I, which is also the nucleation rate, a summation of the balance equations for $I/(B_g n_g^e)$ gives

$$\sum_{g=1}^{G-1} \frac{I}{B_g n_g^e} = U_1 - U_G = 1. \tag{36a}$$

Hence

$$I = \left[\sum_{g=1}^{G-1} \frac{1}{B_g n_g^e}\right]^{-1}. \tag{36b}$$

In numerical calculations, G does not need to greatly exceed g_c for the sum to converge as the omitted terms beyond G in eq. (36b) become entirely negligible. Once I has been calculated, U_g may be found by

$$U_g = I \sum_{g'=g}^{G-1} \frac{1}{B_{g'} n_{g'}^e}. \tag{37}$$

The steady state cluster concentrations n_g are thus given by U_g multiplied by the previously calculated n_g^e.

The occurrence of nucleation reduces the concentration of critical clusters to approximately $n_{g_c} = 0.5 n_{g_c}^e$ at steady state (i.e., $U_{g_c} \approx 0.5$ is found), and clusters somewhat larger than the critical size still have a significant possibility of undergoing evaporation and falling back to or below the critical size (as shown by gradually declining U_g for $g > g_c$). A comparison of the kinetic representation with the classical thermodynamic representation for nucleation shows that the Zeldovich factor is given by

$$Z = U_{g_c} - U_{g_c+1}. \tag{38}$$

The predicted nucleation rates are quite similar for either the standard thermodynamic formulation or the detailed kinetic formulation of classical nucleation. They differ because of subtle variations in the treatment of the smallest clusters, where the simplifying assumptions (such as spherical shape and the use of macroscopic properties) are not very good anyway. The kinetic formulation gives pseudoequilibrium cluster concentrations and thus a rate of nucleation higher by a factor of $S^{\Theta-1}$ than the original thermodynamic form of classical nucleation. (This factor typically amounts to three orders of magnitude, based on water vapor at ambient temperatures and a nucleation rate of roughly 1/cm^3/sec.) The kinetic model uses the Kelvin relationship down to sizes for which it departs slightly from the Gibbs–Thompson expression, but both are of questionable validity at such a small size (which typically reduces the kinetic model nucleation rate by a factor of 3 from the classical thermodynamic). This kinetic model also includes an enhancement factor for the collision frequency factor B_g by considering the thermal motion of the g-mer and finite collision radius of the monomer (which typically seem to increase the rate of nucleation by a factor of 6 over the classical thermodynamic theory). Realistically, however, classical nucleation theory includes several extrapolations of macroscopic phenomena to a microscopic system so that these differences—typically amounting to three orders of magnitude for rate—are not very important; the competing Lothe–Pound theory of nucleation predicts water vapor nucleation rates that are about 17 orders of magnitude higher than classical predictions. The extremely high dependence of nucleation rate on saturation ratio and surface tension makes it difficult experimentally to distinguish predictions differing by many orders of magnitude.

Cluster Scavenging By an Aerosol

Classical nucleation theory does not include the possibility of the scavenging of clusters by existing particles. As Carlton (1980), Friedlander (1982), and McGraw and Marlow (1983) have shown, this addition may be accomplished through a removal term added to the differential equation for cluster concentrations,

$$\frac{dn_g}{dt} = I_{g-1} - I_g - L_g, \tag{39}$$

where L_g is the rate of scavenging of g-mers by aerosol.

In the simplest model for cluster scavenging (as used in all the previous studies just mentioned), it is assumed that the aerosol is in the free molecule size regime, yet consists of particles much larger than the critical cluster size, so the aerosol diffusivity can be neglected when compared to monomer and cluster diffusivity. Adapting Eqs. (27)–(29) for a g-mer instead of a monomer, scavenged by aerosol instead of clusters, gives, for a free molecule regime aerosol,

$$L_g = \frac{\bar{c}_1 A}{4g^{0.5}} n_g, \tag{40}$$

where A is defined as the total aerosol surface area concentration.

A more general expression for the rate of scavenging L_g can be determined if the size distribution is known by taking an integral of the collision function for a g-mer with an aerosol particle multiplied by the actual aerosol distribution over the range of aerosol particle sizes

$$L_g = n_g \hat{C}_g, \tag{41a}$$

where

$$\hat{C}_g = \int_0^\infty \beta_{d_g d_p} \frac{dn(d_p)}{dd_p}\, dd_p. \tag{41b}$$

Although this would involve much numerical effort in any application with a changing aerosol distribution, it is possible to approximate Eq. (41b) by using the Eq. (9) integral for condensation of the monomer onto the aerosol and scaling by the dependence of the scavenging rate on cluster size. For free molecule regime aerosol, this dependence of $\hat{C}_g$ on g is simply $g^{-0.5}$, as shown in Eq. (40). This approach is attractive for a model combining nucleation and condensation, as the rate of condensation will be computed each time the nucleation rate is. For any aerosol distribution, the total net mass condensation rate R'_c may be obtained by Eq. (9) if the particle distribution and vapor saturation ratio are known (in addition to the appropriate physical constants).

If R'_{c+} is calculated, an effective aerosol surface area, A_E, may be obtained from Eq. (10), and used in Eq. (40) replacing A, since

$$L_1 = \frac{R'_{c+}}{m_1} = \frac{\bar{c}_1 A_E n_1}{4}. \tag{42}$$

At steady state the population balance for clusters is now

$$\frac{dn_g}{dt} = B_{g-1}n_{g-1} - E_g n_g - B_g n_g + E_{g+1}n_{g+1} - \hat{C}_g n_g = 0. \tag{43}$$

Again the pseudoequilibrium state (in the absence of aerosol) n_g^e can simplify calculations, especially if n_g^e and P_g have already been calculated, as

$$\frac{dn_g}{dt} = B_{g-1}n_{g-1}^e U_{g-1} - E_g n_g^e U_g - B_g n_g^e U_g + E_{g+1}n_{g+1}^e U_{g+1} - \hat{C}_g n_g^e U_g = 0 \tag{44}$$

$$P_{g-1}\left[U_{g-1} - U_g\right] - P_g\left[U_g - U_{g+1}\right] - C_g U_g = 0 \tag{45}$$

where, for aerosol in any size regime,

$$C_g = \hat{C}_g n_g^e = \frac{\bar{c}_1}{4g^{0.5}} A_E n_g^e. \tag{46}$$

The boundary conditions are identical to those in the aerosol-free case, $U_1 = 1$ and $U_G = 0$. Hence we have a tridiagonal system of G equations in terms of U_g, a dimensionless concentration. One way to proceed is to solve for the ratio U_{g-1}/U_g, which we call $\hat{U}_g$,

$$\hat{U}_g = \frac{U_{g-1}}{U_g} = \frac{C_g + P_g + P_{g-1}}{P_{g-1}} - \frac{P_g}{P_{g-1}}\frac{1}{\hat{U}_{g+1}}. \tag{47}$$

The Szilard boundary condition may be applied at $g = G - 1$, so the last term of Eq. (47) goes to zero,[1] and the system of equations for U_g may be solved by direct substitution back to $g = 2$. Equation (35) allows the droplet current to be computed for any g for the scavenged U_g distribution. Hence the nucleation rate depends on the (smallest) particle size of interest, since, unlike the aerosol-free case, I is a function of g, even for g larger than g_c.

A SIMPLE DYNAMIC NUCLEATION-CONDENSATION MODEL

The simplest useful dynamic model for a system undergoing nucleation must be able to predict nucleation rate with time. Nucleation is a function of the saturation ratio, and a number of physical properties that remain unchanged with time for a simple constant-rate aerosol reactor. (For the moment we shall assume that classical nucleation theory applies. A sufficient aerosol concentration can alter the cluster distribution and hence the nucleation rate, even given the same saturation ratio. Classical nucleation theory neglects the effect of existing aerosol, unless the particles alter the saturation ratio.) The dependence of classical nucleation rate on basic parameters is, from Eq. (23),

$$J = S^2\left[\frac{p_0}{kT}\right]^2 \frac{s_1}{3\pi}\left[\frac{kT\Theta}{2m_1}\right]^{0.5} e^{-(4\Theta^3/27\ln^2 S)}. \tag{48}$$

For a constant-rate aerosol reactor, the only time-dependent parameter in Eq. (48) is S. A dynamic expression for S depends on the mass rates of source generation (R'_s), nucleation (J'), and condensation (R'_c), and

[1]The Szilard boundary condition sets $\hat{U}_G = 0$. More realistically, the second term equal to $[B_g/B_{g-1}][n_g^{e2}/(n_{g-1}^e n_{g+1}^e)][n_{g+1}/n_g]$ will not abruptly go to zero at $g = G - 1$. Instead, for a G sufficiently large, each term inside brackets will quite rapidly approach unity. Hence, when there is g-mer scavenging by aerosol, the boundary condition $\hat{U}_g = (C_g + P_g)/P_{g-1}$ at $g = G - 1$ is an improvement over the Szilard boundary condition, allowing U_g values to be highly accurate for g closer to the $G - 1$ approximate boundary condition.

may be written

$$\frac{dS}{dt} = \frac{R'_s - J' - R'_c}{m_1 p_0/kT}. \tag{49}$$

For a constant-rate aerosol reactor, R'_s is constant with time. If some (fixed) supercritical size at diameter d_s and cluster number g_s is taken as the size at which new particles form and hence condensation begins, the following expressions apply:

$$J' = Jm_1 g_s \tag{50}$$

$$R'_c = m_1 \frac{p_0}{kT} \frac{\overline{c_1}}{4} \int_{d_s}^{\infty} \left(S - \frac{p_d}{p_0}\right) \times \pi d_p^2 f(Kn) \frac{dN}{dd_p} \, dd_p. \tag{51}$$

Note that d_s should be selected so that for any time when significant rates of nucleation occur, d_s is at least as large as the critical size, yet small enough so that the droplet current through g_s is the same as through g_c, so that steady state nucleation applies into that size range.

The expression for dS/dt includes a condensation term, which depends on the existing aerosol. In its simplest form, where the aerosol is in the free molecular regime, the condensation rate is proportional to the total aerosol surface area; in the general form, the condensation rate must be obtained from an integral of the aerosol size distribution. Even for a free molecule aerosol, the dynamic expression for A cannot be obtained without knowledge of the size distribution. (It is assumed that the bulk mean saturation ratio adequately describes the system, so the concentration profiles around each aerosol particle need not be considered.)

To proceed without a complicated dynamic aerosol model, one may approximate the necessarily polydisperse aerosol by a monodisperse one. (This approximation is intended primarily for the initially aerosol-free system and will also be reasonable when the initial aerosol has a single mode and is concentrated enough to make homogeneous nucleation fairly insignificant.) The (number) mean aerosol mass $\overline{m}_p$ is given by

$$\overline{m}_p = Q/N. \tag{52}$$

The total aerosol mass Q and total aerosol number N (both per unit volume) may be obtained by integrating the following differential equations:

$$\frac{dQ}{dt} = R'_c + J' \tag{53}$$

$$\frac{dN}{dt} = J. \tag{54}$$

From $\overline{m}_p$, a mean particle diameter $\overline{d}_p$ and a mean particle surface area $\overline{s}_p$, are readily obtained, allowing evaluation of R'_c. Hence a well-mixed system undergoing nucleation and condensation may be approximately modeled by a system of three (strongly coupled) ordinary differential equations.

Representing a continuous aerosol size distribution as monodisperse at a characteristic mean size (which varies with time) is a fairly crude approximation. Although the total mass and total number are not altered directly (although the time profile of all the aerosol properties will be somewhat different, because the equations are coupled), the total surface area and number mean diameter necessarily will differ from the actual polydisperse distribution. Interestingly, given fixed total mass and total number, a monodisperse aerosol has the maximum total surface area as well as number–diameter product (or number mean diameter) of any possible aerosol distribution. Since condensation is proportional to a quantity ranging between the total surface area and the number–diameter product, the monodisperse model will systematically overpredict the condensation rate (given S, Q, and N, where S alone is sufficient to fix the nucleation rate), though probably not by more than a factor of 2. (Recall that three orders of magnitude discrepancy between nucleation expressions is considered tolerable, since J is normally such an extremely strong function of S.) Insomuch as nucleation and condensation are competing processes, the overprediction of condensation rates is similar to underpredicting nucleation rates, and the monodisperse model should somewhat underpredict the total number of particles produced according to classical nucleation with an accurate aerosol size representation.

This monodisperse model can also accommodate a nucleation expression that includes the effect of aerosol on the nucleation rate because of cluster scavenging, giving J' a dependence on N and on $\overline{m}_p$ as well as on S.

Even this very simple dynamic model of nucleation and condensation addresses some important questions that are currently unanswered for systems undergoing a burst of nucleation. They include the following:

1. How many particles are formed by nucleation?
2. What is the mean size of the particles once nucleation ceases?
3. How long will the burst of nucleation last?
4. How does the total number of particles formed by nucleation depend on the rate of monomer generation and on physical properties of the vapor, such as equilibrium vapor pressure and surface tension? What are the relevant dimensionless groups?

DIMENSIONLESS PARAMETERS

Systems undergoing simultaneous nucleation and condensation can be conveniently characterized in terms of a few nondimensional parameters, some time-invariant and others changing as the aerosol evolves.

The nucleation equations are readily expressed in dimensionless form. In the absence of pre-existing aerosol, the dimensionless steady state nucleation rate $\hat{J}$ is a function only of the saturation ratio S and a surface energy factor Θ, both already dimensionless. The characteristic concentration is n_{sat}, the concentration of monomer vapor at saturation. A characteristic collision rate, R_β, is proportional (and nearly equal) to the rate of monomer–monomer collisions in the saturated vapor. The dimensionless classical expression for nucleation is thus

$$\hat{J} = \frac{J}{\hat{R}_\beta} = g_c^{2/3} S^2 Z e^{-W_c} \tag{55}$$

$$\hat{R}_\beta = \frac{n_{\text{sat}}^2 \overline{c}_1 s_1}{4}. \tag{56}$$

W_c and g_c are given by Eq. (18). The kinetic form of classical nucleation may be likewise presented in nondimensional form by scaling with $\hat{R}_\beta$ and n_{sat}.

The presence of an aerosol only adds one additional dimensionless group to the model,

$$\tilde{A} = \frac{A}{n_{\text{sat}} s_1} \tag{57}$$

where $\tilde{A}$ is a dimensionless aerosol surface area, formulated as the ratio of aerosol surface area to surface area of the saturated monomer. In more general form, one may replace $\tilde{A}$ with $\tilde{A}_E$, a dimensionless equivalent-free-molecule surface area. $\tilde{A}_E$ is a rate ratio between the rate of monomer collisions with aerosol surface and the rate of monomer colliding with monomer (neglecting an enhancement factor), calculated for the saturated state so as to be independent of S, and thus, from Eqs. (42) and (56),

$$\tilde{A}_E = \frac{R'_{c+}}{m_1 S \hat{R}_\beta}. \tag{58}$$

Since monomer–aerosol collisions increase with S whereas monomer–monomer collisions increase as S^2, the ratio of collisions with aerosol to collisions with monomer is proportional to $\tilde{A}_E/S$ for the monomer or cluster. There is a further size dependence, which ultimately gives, for a g-mer,

$$R_g = \frac{\text{Collision rate with aerosol}}{\text{Collision rate with monomer}} = \frac{\tilde{A}_E}{S(1+g)^{0.5}(1+g^{1/3})^2}. \tag{59}$$

For a cluster with $g \gg 1$ yet still in the free molecule regime, this simplifies to

$$R_g \approx \frac{\tilde{A}}{Sg^{7/6}}. \tag{60}$$

Nondimensionally, the aerosol scavenged population balance equation will have the same form as the dimensional equation. U_g and $\hat{U}_g$ are already dimensionless; n_g^e, P_g, and C_g may be replaced, respectively, by $\tilde{N}_g$, $\tilde{P}_g$ and $\tilde{C}_g$, where

$$\tilde{N}_g = \frac{n_g^e}{n_{\text{sat}}} \tag{61}$$

$$\tilde{P}_g = \frac{P_g}{\hat{R}_\beta} = S\tilde{N}_g(1 + 1/g)^{0.5}(1 + g^{1/3})^2 \tag{62}$$

$$\tilde{C}_g = \frac{C_g}{\hat{R}_\beta} = \frac{\tilde{A}\tilde{N}_g}{g^{0.5}} = R_g\tilde{P}_g. \tag{63}$$

This allows one to write the following:

$$\hat{U}_g = 1 + \frac{\tilde{P}_g}{\tilde{P}_{g-1}}\left(1 + R_g - \frac{1}{\hat{U}_{g-1}}\right) \tag{64a}$$

$$\hat{U}_{G-1} = \frac{\tilde{P}_{G-1}}{\tilde{P}_{G-2}}(1 + R_{G-1}). \tag{64b}$$

The solution is entirely analogous to the dimensional case, yielding

$$\hat{J}_g = \tilde{P}_{g-1}(U_{g-1} - U_g) = \tilde{P}_{g-1}U_{g-1}\left(1 - \frac{1}{\hat{U}_g}\right). \tag{65}$$

For the cases considered here, nucleation is driven by a constant source of condensable vapor, R_s' (grams cm^{-3} sec^{-1}). The source rate may alternatively be expressed as a number rate, $R_s = R_s'/m_1$ and be made dimensionless in the form

$$\tilde{R}_s = \frac{R_s}{\hat{R}_\beta}. \tag{66}$$

$\tilde{R}_s$ may be interpreted as the ratio of the time scale for monomer production τ_s to that for monomer collisions τ_β, where the time scales are given by,

$$\tau_s = \frac{n_{\text{sat}}}{R_s}$$
$$= \text{time for source to regenerate saturation concentration} \tag{67}$$

$$\tau_\beta = \frac{n_{\text{sat}}}{\hat{R}_\beta} \approx \frac{S}{B_1}$$
$$\approx \text{time between collisions for saturated monomer.} \tag{68}$$

There is another important intrinsic time scale for nucleation, and this is the time lag, or time constant for the approach to steady state. Various definitions of the time lag have been offered for the approach to steady state, as summarized by Abrahams (1974). There is no simple but accurate way to use homogeneous nucleation theory when the steady state assumption does not apply; a dynamic model for cluster concentrations with time must be solved. The nucleation time lag (for the critical size cluster) is given approximately (using the expression by Collins) as

$$\tau_n = \frac{1}{4B_{g_c}Z^2} \approx \frac{\tau_\beta}{4Z^2g_c^{2/3}S} \approx \frac{\tau_\beta}{S}O(1). \tag{69}$$

Hence τ_n will be of the same order as τ_β (and usually slightly larger). Classical nucleation theory or related nucleation theories (including this cluster scavenging model) require an approximate steady state for subcritical g-mer concentrations, and thus are applicable only when (approximately)

$$\tilde{R}_s \ll S \quad \text{or} \quad \tilde{R}_s < 1. \tag{70}$$

Analysis of the competition between nucleation and condensation for monomer leads to additional dimensionless groups.

It is convenient to define a dimensionless time $\tilde{t}$ based on τ_s,

$$\tilde{t} = t/\tau_s. \tag{71}$$

The properties of the aerosol distribution are also conveniently presented in dimensionless form. In addition to the dimensionless total surface area, $\tilde{A}$, the dimensionless total number $\tilde{N}$ and the dimensionless aerosol mass $\tilde{Q}$

can be defined as follows:

$$\tilde{N} = N/n_{\text{sat}} \tag{72}$$

$$\tilde{Q} = Q/(n_{\text{sat}} m_1). \tag{73}$$

The number mean dimensionless particle size, $\bar{g}_p$, or typical number of monomer units per particle, is given by

$$\bar{g}_p = \tilde{Q}/\tilde{N}. \tag{74}$$

For a system beginning with no vapor or particles initially, $S = \tilde{t}$ until gas-to-particle conversion becomes significant. Thereafter, the mass split for the condensable species between the vapor and aerosol phases is given by the ratio $S : \tilde{Q}$, and

$$S + \tilde{Q} = \tilde{t}. \tag{75}$$

The nucleation rate may be made dimensionless on either a number basis, as $\tilde{J}$, or on a mass basis, as $\tilde{J}'$. Condensation rate is made dimensionless as $\tilde{R}_c$.

$$\tilde{J} = \frac{J}{R_s} = \frac{Jm_1}{R_s'} \tag{76}$$

$$\tilde{J}' = \frac{J'}{R_s'} = \frac{Jm_1 g_s}{R_s'} \tag{77}$$

$$\tilde{R}_c = \frac{R_c'}{R_s'}. \tag{78}$$

Note that for an aerosol undergoing only nucleation and condensation, only homogeneous nucleation changes particle number. And only nucleation and condensation contribute to total aerosol mass. So for a system beginning with no aerosol, at dimensionless time $\tilde{t}$,

$$\tilde{N} = \int_0^{\tilde{t}} \tilde{J}\, d\tilde{t}' \tag{79}$$

$$\tilde{Q} = \int_0^{\tilde{t}} (\tilde{J}' + \tilde{R}_c)\, d\tilde{t}' \tag{80}$$

$$S = \int_0^{\tilde{t}} (1 - \tilde{J}' - \tilde{R}_c)\, d\tilde{t}'. \tag{81}$$

Assuming that nucleation and condensation are the dominant processes during early aerosol evolution, the aerosol number concentration N will reach a constant value, denoted by N_T, once homogeneous nucleation is over. This value, or the dimensionless final number concentration $\tilde{N}_T$, which $\tilde{N}$ goes to, is a very useful measure of the overall amount of nucleation that occurs in a system. (In the case of a system with initial aerosol, it is the change in N or $\tilde{N}$ that measures the importance of homogeneous nucleation.)

Three more dimensionless parameters can be defined to trace the importance of competing processes—nucleation, condensation, and cluster scavenging by aerosol—in the gas-to-particle conversion process:

$$\varphi_1 = \frac{J_{\text{scav}}}{J_{\text{unscav}}} \tag{82}$$

$$\varphi_2 = \frac{\tilde{J}'}{\tilde{J}' + \tilde{R}_c} = \frac{J'}{J' + R_c'} \tag{83}$$

$$\varphi_3 = \tilde{J}' + \tilde{R}_c = \frac{J' + R_c'}{R_s'} \tag{84}$$

φ_1 gives the instantaneous effect of the aerosol on the nucleation rate due to cluster scavenging. ($\varphi_1 = 1$ if no effect and $\varphi_1 = 0$ if cluster scavenging totally eliminates nucleation.) φ_2 tells what fraction of the mass going to particles is going by homogeneous nucleation to the creation of very small particles. φ_3 shows the efficiency at which generated vapor is being converted to aerosol mass.

SIMULATION OF NUCLEATION AND GROWTH FROM A CONTINUOUSLY REINFORCED VAPOR

The simple monodisperse aerosol model developed above for nucleation and condensation was used with a variety of physical parameters for the different expressions for homogeneous nucleation to simulate aerosol evolution from a continuously reinforced vapor. A standard test case was considered for a hypothetical low vapor pressure organic species having physical properties as listed in Table 1. The predictions made by the monodisperse model were compared with those

TABLE 1. Physical Properties of the Model Compound

Property	Symbol	Value	Units
Temperature	T	298	K
Total pressure	p	1	atm
Molecular weight	M_1	100	g gmole^{-1}
Liquid density	ρ_l	1	g cm^{-3}
Diffusivity	D_1	0.0411	cm^2 sec^{-1}
Surface tension[a]	σ	25	dyne cm^{-1}
Vapor pressure[b]	p_0	0.00001	dyne cm^{-2}
Dimensionless surface energy[a]	Θ	8.878	
Characteristic collision time[c]	τ_β	44.8	sec

[a]Varied for Figure 8.
[b]Value does not affect dimensionless results.
[c]Is inversely proportional to p_0.

from a sectional dynamic aerosol model, as described by Gelbard and Seinfeld (1980), modified to include homogeneous nucleation into the smallest aerosol size section.

The first five figures show the transient dimensionless system parameters as functions of dimensionless time for different dimensionless source rates and different models. All cases consider only homogeneous nucleation and condensation with a constant source of monomer and no initial aerosol. The first three figures show the results of the monodisperse aerosol model assuming the standard classical nucleation expression. Figures 1–3 exhibit results for dimensionless source rates of 10^{-2}, 10^{-4}, and 1, respectively, the latter value being approximately that where all steady state

FIGURE 1. Aerosol evolution for a dimensionless source rate $\tilde{R}_s = 10^{-2}$ assuming standard classical nucleation theory and the monodisperse aerosol model.

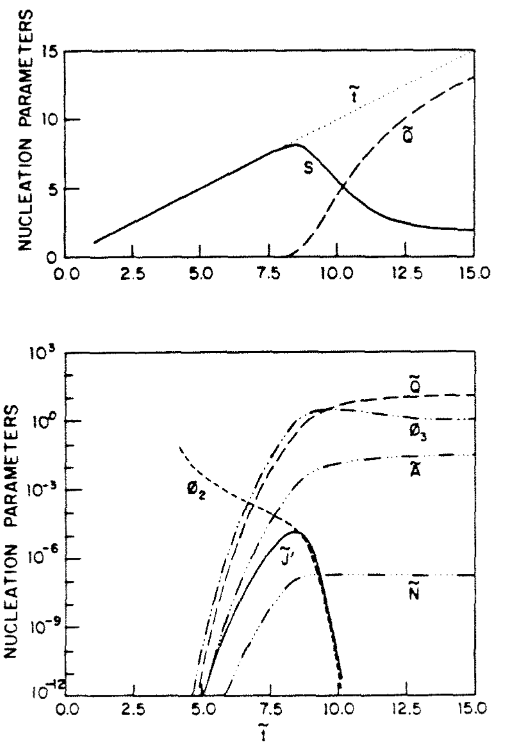
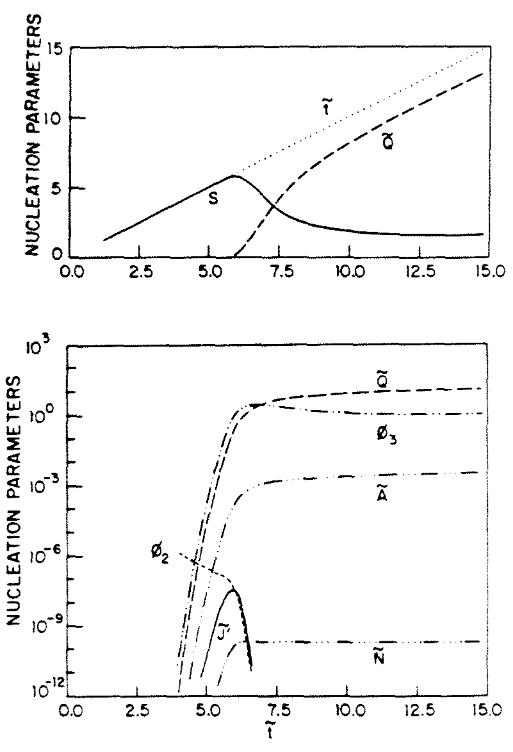


FIGURE 2. Aerosol evolution for a dimensionless source rate $\tilde{R}_s = 10^{-4}$ assuming standard classical nucleation theory and the monodisperse aerosol model.



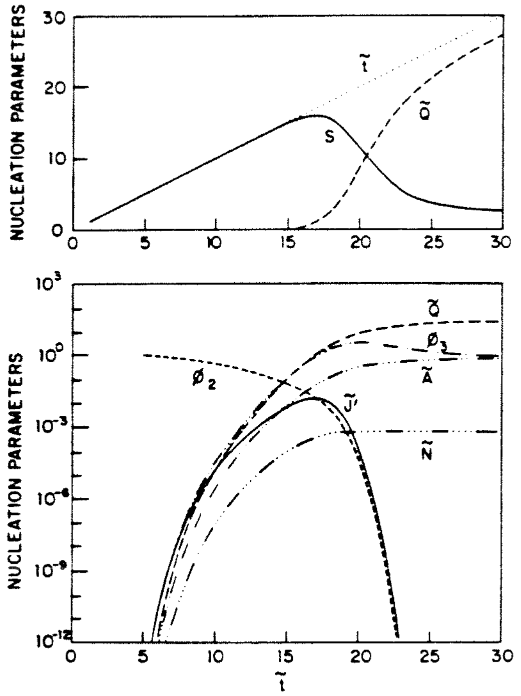


FIGURE 3. Aerosol evolution for a dimensionless source rate $\tilde{R}_s = 1$ assuming standard classical nucleation theory and the monodisperse aerosol model.

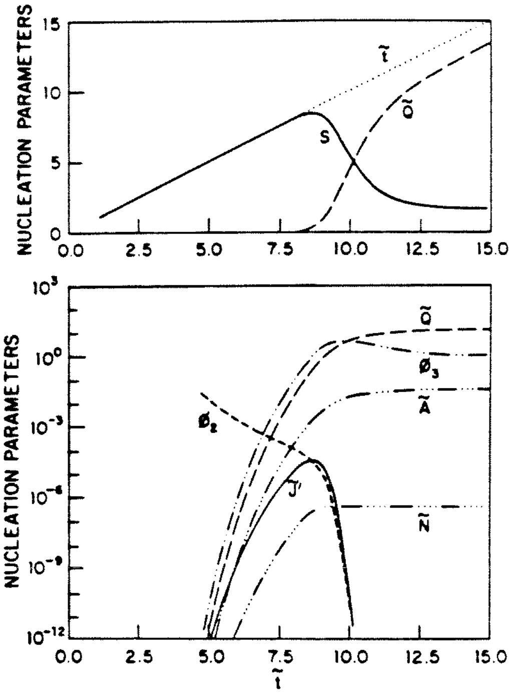


FIGURE 4. Aerosol evolution for a dimensionless source rate $\tilde{R}_s = 10^{-2}$ assuming standard classical nucleation theory and the sectional aerosol model.

nucleation models reach the limit of their applicability.

Figure 4 shows the nucleation parameters for a dimensionless source rate of 0.01, using the size-resolved sectional dynamic aerosol model, with the coagulation and deposition processes omitted, assuming the standard classical nucleation expression. The resulting particle number concentration is larger by a factor of two than predicted by the simple monodisperse model, due to lower predicted overall condensation rates.

Figure 5 shows the nucleation parameters for a dimensionless source rate of 0.01, using the kinetic cluster balance form of classical nucleation with the monodisperse aerosol model. Also, cluster scavenging by aerosol is included—though, since the φ_1 curve was found to stay virtually at one, the effect is totally negligible. Note that the Figure 5 results are quite similar to those of Figure 1, where the standard closed classical nucleation expression was implemented; the total number concentration is higher by a factor of about three for the expanded cluster balance nucleation expression.

Although the five time profiles differ in numerical values, their behavior is qualitatively very similar. The saturation ratio (S) is equal to dimensionless time ($\tilde{t}$) almost up until the peak of dimensionless nucleation ($\tilde{J}'$), whereupon the saturation ratio falls off drastically, asymptotically approaching unity with large time. The peak saturation ratio (around 5 to 15, increasing with dimensionless source rate) coincides with the nucleation rate peak, which also coincides with the gas-to-particle conversion efficiency curve (φ_3) rising through unity. Interestingly, although nucleation of very small particles

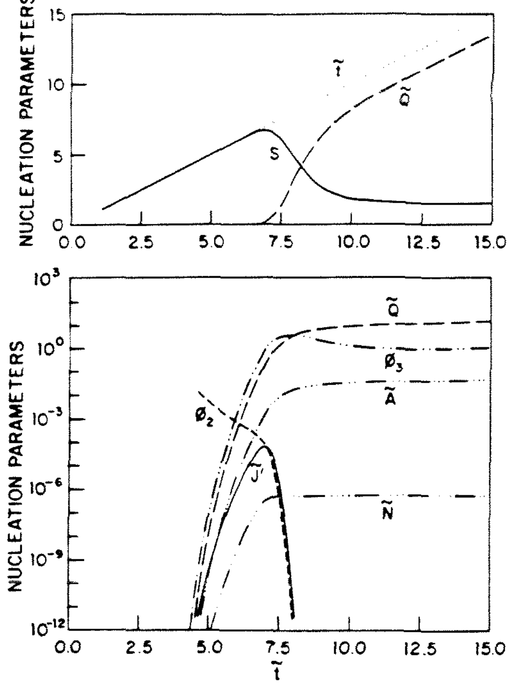


FIGURE 5. Aerosol evolution or a dimensionless source rate $\tilde{R}_s = 10^{-2}$ assuming cluster-scavenged nucleation theory and the monodisperse aerosol model.

dominates the gas-to-particle conversion process (as measured by φ_2) as it must at the onset of nucleation, for significant rates of nucleation the vast majority of the mass forming aerosol does so by condensation onto the nucleated particles, not by nucleation of new particles. At peak nucleation rates, the fraction of the mass going to aerosol by homogeneous nucleation is no more than 1% at the highest dimensionless source rate, and this fraction φ_2 has more than a first order dependence on the source rate. Hence condensation is strongly favored over nucleation, once nucleation has provided particles on which condensation may occur.

The mean number of monomers per aerosol particle ($\bar{g}_p$) is equal to the dimensionless total mass ($\tilde{Q}$) divided by the dimensionless total number ($\tilde{N}$) (both being scaled to the saturated monomer state). Thus $\bar{g}_p$ goes as the reciprocal of φ_2 (over a particle number average), and typically ranges from hundreds to billions, going inversely with $\tilde{R}_s$ to a power greater than one. Since the average particle formed is tens or hundreds of monomer diameters in size, and the dimensionless aerosol mass is of order one to ten (similar to the peak saturation ratio) during the burst of nucleation.the dimensionless aerosol surface area will be less than order one, and cluster scavenging by the aerosol should not be significant. This is confirmed by the φ_1 profiles (not shown), which remain very near unity (indicating no effect by cluster scavenging) unless $\tilde{A}_E$ is at least order one, which occurs only when $\tilde{R}_s$ is at least order one. For the $\tilde{R}_s = 1$ case, only after the nucleation peak does φ_1 fall, so the overall effect was still small.

Although no results with pre-existing particles will be presented here, an interesting observation can be made. If a burst of homogeneous nucleation occurs, S will peak and decline, and φ_3 will overshoot one, with peak nucleation as φ_3 crosses one. For steady state nucleation, φ_2 is much less than unity at the nucleation rate peak, so condensation is much greater than nucleation, and $R_c' = R_s'$ is approximately true at the moment of peak nucleation and peak supersaturation. By Eq. (13), neglecting the Kelvin effect, the peak saturation ratio should be given by

$$S_{\max} = 1 + \frac{4R_c'}{m_1\bar{c}_1 n_{\text{sat}} A_E}. \tag{85}$$

Assuming $R_c' = R_s'$ and converting to nondimensional form yields

$$S_{\max} = 1 + \frac{\tilde{R}_s}{\tilde{A}_E}. \tag{86}$$

Since condensation and nucleation can only increase $\tilde{A}_E$, a sufficient condition for existing aerosol to completely inhibit homogeneous nucleation may be calculated. (However, the neglected processes of coagulation or surface deposition could decrease $\tilde{A}_E$ and lead to a slow rate of homogeneous nucleation in the long term.) Assume that a noticeable rate of homogeneous nucleation occurs for $S \geq S_{\text{crit}}$, which can be calculated for a system. If the initial aerosol distribution (or the evolving aerosol distribution until at least $\tilde{t} = S_{\text{crit}}$, or so long as $S < S_{\text{crit}}$) has dimensionless equivalent surface area above the following bound, homogeneous nucleation will not occur:

$$\tilde{A}_E \geq \frac{\tilde{R}_s}{S_{\text{crit}} - 1}. \tag{87}$$

We can now make definitive conclusions regarding the influence of cluster scavenging by the aerosol. As previously found by Carlton (1980), Friedlander (1982), and McGraw and Marlow (1983), the nucleation rate (for fixed S) is appreciably lowered by cluster scavenging when the dimensionless aerosol concentration appreciably exceeds unity. However, when nucleation occurs in an initially aerosol-free system, the inclusion of cluster scavenging in the nucleation expression has virtually no effect on the evolving aerosol distribution. Essentially all the nucleation will occur before the aerosol concentration is high enough to perturb the cluster distribution significantly, although the aerosol concentration becomes high enough to deplete significantly the monomer concentration by condensation. Cluster scavenging is important in an initially particle-free system only when the dimensionless source rate exceeds unity. But this is also the point at which all steady state nucleation expressions become suspect for a variety of reasons. Among them are the problem of the nucleation time [Eq. (69)], and the increasing numbers of clusters and aerosol particles relative to monomer. The nucleation theories assume that monomer concentrations are sufficient to overwhelm the effects of cluster–cluster collisions and aerosol coagulation, during the nucleation period, which does not hold for dimensionless source rates much greater than unity, which produce very large supersaturation ratios and numbers of particles. With a dimensionless source rate of one, the total number of particles produced by nucleation is apparently decreased by about 1 or 2% due to cluster scavenging. Figure 6 shows the reduction in resulting particle number concentration caused by including cluster scavenging in the kinetic population balance (using the monodisperse aerosol model, standard test case), for different dimensionless source rates.

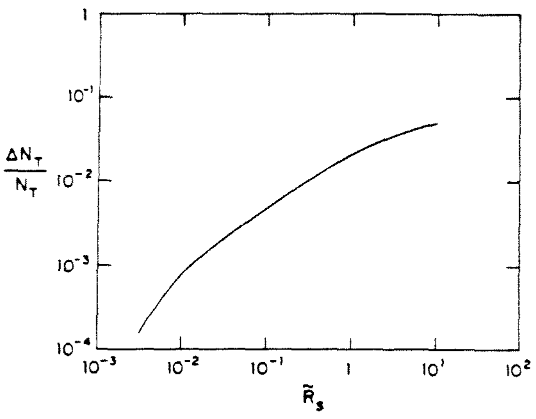


FIGURE 6. Fractional reduction in the number of new particles nucleated due to cluster scavenging by the aerosol, as a function of dimensionless source rate $\tilde{R}_s$.

The inclusion of pre-existing aerosol in the system does not alter the finding that classical nucleation theory does not need to be modified to include cluster scavenging. If the pre-existing aerosol is concentrated enough to result in a dimensionless surface area of order unity or greater, one of two effects will nullify the utility of considering cluster-scavenging. Often condensation onto the existing aerosol will limit the supersaturation and make the rate of homogeneous nucleation completely negligible even without considering cluster scavenging, as shown by Eq. (87). Otherwise, the dimensionless source rate of vapor generation will be so high that no steady state nucleation expression can be considered valid, because the cluster profile will not remain in steady state with the changing monomer concentrations. Using Eq. (87), if a significant rate of

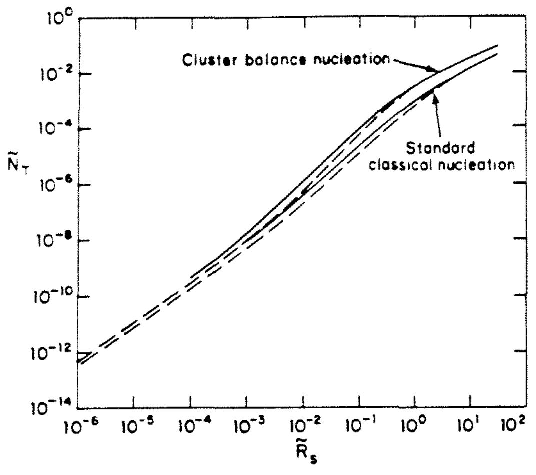


FIGURE 7. Resulting dimensionless aerosol concentration $\tilde{N}_T$ as a function of dimensionless source rate $\tilde{R}_s$ for the model compound using alternate aerosol and nucleation models (solid line, sectional aerosol model; broken line, monodisperse aerosol model).

homogeneous nucleation is to occur,

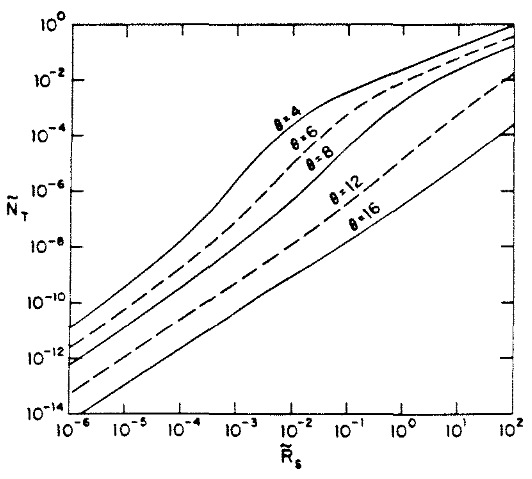
$$\tilde{R}_s \geq S_{\text{crit}} - 1 \geq 1 \qquad \text{whenever} \quad \tilde{A}_E \geq 1. \tag{88}$$

Hence both the dimensionless source rate and the dimensionless effective surface area must be greater than unity for cluster scavenging by the aerosol to be significant. And this case cannot be handled accurately by any nucleation expression assuming a steady state cluster profile.

Figure 7 shows the dependence of the final dimensionless aerosol number concentration $\tilde{N}_T$ resulting from nucleation (there was no initial aerosol) on the dimensionless rate of source generation $\tilde{R}_s$ for the standard test case, assuming either the standard closed (thermodynamic) expression for classical nucleation or the kinetic cluster balance form of classical nucleation. The monodisperse model gives slightly higher number concentrations (as expected) than the complete sectional model, but only by a factor of about two over a range of 12 orders of magnitude in resulting particle number concentrations. The kinetic cluster balance formulation of classical nucleation predicts from about 1.4 ($\tilde{R}_s = 10^{-6}$) to 4 ($\tilde{R}_s = 1$) times as many particles as the standard classical nucleation expression.

It should be noted that the dimensionless results are independent of vapor pressure, which is involved in nondimensionalizing the source rate. If the vapor pressure is doubled, the vapor source rate must be quadrupled to have the same dimensionless source rate $\tilde{R}_s$; the particle number concentration resulting from the burst of nucleation will also be doubled, yielding the same dimensionless total nucleation, which is scaled to the saturated monomer concentration. Still, Figure 7 is not a master curve for homogeneous nucleation. Resulting number concentration is also a strong function of the dimensionless surface energy, as show in Figure 8. (Figure 8 shows $\tilde{N}_T$ for different values of Θ, using the monodisperse aerosol model with the standard classical nucleation expression; the surface tension was varied to obtain different Θ.) To a lesser extent, the resulting dimensionless number concentration also depends

FIGURE 8. Resulting dimensionless aerosol concentration $\tilde{N}_T$ as a function of dimensionless source rate $\tilde{R}_s$ for various dimensionless surface energies Θ assuming standard classical nucleation theory and the monodisperse aerosol model.



on the other groups involving monomer diffusivity, monomer Knudsen number, and monomer molecular weight.

The curves in Figures 7 and 8 show three distinct regions. For low dimensionless source rates ($\tilde{R}_s < 0.01$), the total number of particles nucleated goes as the source rate to about the 1.38 power, and freshly nucleated particles very rapidly reach the transition or continuum size regimes. For dimensionless source rates of about 0.01 to 1, the total number of particles nucleated is a stronger function of $\tilde{R}_s$, going as high as the square. For still higher source rates, where steady state nucleation no longer applies, the resulting total number by nucleation seems to assume a weaker dependence on $\tilde{R}_s$, falling below first power.

CONCLUSIONS

For a relatively particle-free system with a constant source of condensable vapor, the initially increasing supersaturation will lead to a burst of homogeneous nucleation. The new particles will grow rapidly by condensation of the vapor, thus quenching new particle formation by reducing the supersaturation. The resulting number and size of particles depend strongly on the dimensionless vapor source rate $\tilde{R}_s$, which is proportional to the mass generation rate of vapor and inversely proportional to the square of the vapor pressure, as well as the dimensionless surface energy Θ, which is proportional to the surface tension. Particle number concentration shows a dependence of approximately $\tilde{R}_s^{1.4}$ for low $\tilde{R}_s (\leq 10^{-3})$, rising to a peak dependence of approximately $\tilde{R}_s^2$ for intermediate $\tilde{R}_s$ ($\approx 10^{-2}$), where the aerosol lies in the free molecule regime during the nucleation period. For $\tilde{R}_s \leq 1$, even at the peak rate of homogeneous nucleation, the vast majority of gas-to-particle conversion on a mass basis occurs by condensation onto nucleated (or pre-existing) particles that have grown well above the critical size, rather than by the homogeneous nucleation of new particles just above the critical size. Particle number concentrations decrease with increasing Θ, on which the energy barrier to homogeneous nucleation depends. Mean particle mass varies inversely with the aerosol number concentration, and increases nearly lineraly with time once homogeneous nucleation ceases.

For high dimensionless source rates, $\tilde{R}_s \gg 1$, number concentrations appear to rise with less than a first-order dependence on $\tilde{R}_s$, even using classical nucleation expressions. Also, when $\tilde{R}_s \gg 1$, cluster scavenging by the aerosol seems to significantly reduce the nucleation rate beneath classical predictions, as the dimensionless aerosol surface area exceeds unity. Unfortunately, when $\tilde{R}_s \gg 1$, all steady state models of nucleation break down, as the cluster population will not be in steady state with the monomer nor with the monomer and aerosol. Additionally, the saturation ratio rises so high and the critical cluster size falls so low that the neglected cluster–cluster interactions become significant, as may coagulation of the aerosol because of the high number concentrations. Under such conditions, a dynamic model including coagulation and evaporation over the monomer, cluster and aerosol size spectrum appears necessary for accurate aerosol modeling.

This work was supported by National Science Foundation Grant ATM-8208625.

REFERENCES

Abrahams, F. F. (1974). *Homogeneous Nucleation Theory*. Academic Press, New York.

Carlton, G. N. (1980). Predicting new particle formation from preexisting aerosol concentrations. M.S. thesis, University of California, Los Angeles.

Friedlander, S. K. (1982). The behavior of constant rate aerosol reactors. *Aerosol Sci. Technol.* 1:3–13.

Fuchs, N. A., and Sutugin, A. G. (1971). High-dispersed aerosols, in *Topics in Current Aerosol Research*. Hidy, G. M. and Brock, J. R., (eds.). Pergamon, Oxford, Vol. 2, p. 34.

Gelbard, F., and Seinfeld, J. H. (1980). Simulation of multicomponent aerosol dynamics. *J. Colloid Interface Sci.* 78:485–501.

McGraw, R., and Marlow, W. H. (1983). The multistate kinetics of nucleation in the presence of an aerosol. *J. Chem. Phys.* 78:2542–2548.

Pesthy, A. J., Flagan, R. C., and Seinfeld, J. H. (1983). Theory of aerosol formation and growth in laminar flow. *J. Colloid Interface Sci.* 91:525–545.

Received 1 July 1983; accepted 9 November 1983

CHAPTER 4:

PREDICTION OF AEROSOL CONCENTRATIONS
RESULTING FROM A BURST OF NUCLEATION

Published in
Journal of Colloid and Interface Science
105:136–142 (May 1985)

Prediction of Aerosol Concentrations Resulting from a Burst of Nucleation

DALE R. WARREN AND JOHN H. SEINFELD

Department of Chemical Engineering, California Institute of Technology, Pasadena, California 91125

Received June 6, 1984; accepted October 10, 1984

The number of particles formed by a burst of homogeneous nucleation in a closed system is predicted approximately and found to depend, in the absence of initial aerosol, only on the dimensionless source rate of vapor, a dimensionless surface tension, and an appropriately defined Knudsen number. The effect of a seed aerosol on the ultimate number of particles formed is also studied. For the purposes of exploring the nature of systems in which a competition for vapor exists between nucleation and condensation, a closed system with a steady source of vapor is considered. It is shown that a narrow window exists within which seed particles can be used to control the ultimate number of particles formed in such a system. © 1985 Academic Press, Inc.

INTRODUCTION

A key question in the analysis of systems in which new particle formation is occurring by nucleation is what is the ultimate number concentration of particles formed. Since condensable vapor can either nucleate to form new particles or condense on newly formed or preexisting particles, the ultimate number of particles formed is governed by the competition between the nucleation and condensation processes. We have previously studied this competition using models based on the general dynamic equation for aerosols (1, 2). It turns out that a simplified, approximate model can be developed that exhibits all of the qualitative features of the more detailed treatments with respect to the intricate interplay between nucleation and condensation. The purpose of this paper is to present that model and explore its predictions.

We shall consider a system consisting of a condensable vapor and aerosol. The system state is given by the temperature T, total pressure P, the vapor concentration (either number concentration N_v or mass concentration $M_v = m_1 N_v$), and the aerosol size distribution. For simplicity, the aerosol size distribution may be roughly described by its first two moments, the total aerosol number concentration N_p, and the total aerosol mass concentration M_p. We assume that the rates of three processes are of interest: vapor generation R_G, homogeneous nucleation R_J, and condensation R_C, which may be expressed in terms of number per unit volume per unit time. Mass and number balances for the system yield the following:

$$\frac{d}{dt} N_v = R_G - g_s R_J - R_C \qquad [1]$$

$$\frac{d}{dt} N_p = R_J \qquad [2]$$

$$\frac{d}{dt} M_p = m_1 g_s R_J + m_1 R_C. \qquad [3]$$

The somewhat arbitrary diameter at which nucleated clusters are considered to be particles will be denoted by d_s, corresponding to g_s molecules. The exact value of g_s is not crucial, so long as it is slightly above the largest critical number at which significant nucleation can occur, and thus each freshly nucleated particle continues to grow spontaneously. (Near or below the critical size, the net growth or droplet current is dependent upon the nonequilibrium cluster distribution and the forward and reverse growth rates, rather than simply upon the net growth rate for particles of the given size. The expression for condensation rate R_C will be based on net growth rates.)

A total mass balance for this system gives $d(M_v + M_p)/dt = m_1 R_G$. The value of the vapor source rate R_G is assumed to be known, and, for simplicity, constant. The values of R_J and R_C may be calculated from classical nucleation theory and particle mass transfer theory, respectively.

NUCLEATION AND GROWTH RATES

Classical homogeneous nucleation theory expresses nucleation rate (cm^{-3} s^{-1}) as (3)

$$R_J = \left[\frac{\zeta \pi d_c^2 N_v}{\sqrt{2\pi m_1/kT}}\right]\left[\frac{2v_1}{\pi d_c^2}\sqrt{\frac{\sigma}{kT}}\right] \times \left[N_v \exp\left(-\frac{\pi d_c^2 \sigma}{3kT}\right)\right], \quad [4]$$

where the critical diameter $d_c = 4\sigma v_1/kT \times \ln S$, σ is the surface tension, and v_1 is the molecular volume of the nucleating species. Using the saturation ratio $S = N_v/N_s$, where N_s is the saturated vapor number concentration, and assuming an accommodation coefficient ζ of unity,

$$R_J = S^2 N_s^2 2v_1 \sqrt{\frac{\sigma}{2\pi m_1}} \times \exp\left(-\frac{16\pi\sigma^3 v_1^2}{3k^3T^3\ln^2 S}\right). \quad [5]$$

The rate of condensation is a sum of the rate of condensation onto all particles contained within the system volume. For a particle of diameter d_p, the rate at which molecules of vapor condense onto it is given by

$$R_{C_p} = N_s \frac{\bar{c}_1}{4} \pi d_p^2 (S - e^{d_K/d_p}) f(Kn), \quad [6]$$

where the mean molecular speed $\bar{c}_1 = \sqrt{8kT/\pi m_1}$. The characteristic Kelvin diameter d_K equals $2\pi d_1^3\sigma/3kT$, where d_1 is the apparent molecular diameter (for a sphere of volume v_1). The size regime interpolation function $f(Kn)$ is defined to go to unity in the kinetic limit where $Kn \rightarrow \infty$. The Knudsen number Kn is defined by

$$Kn = \frac{2\lambda_1}{d_p}\frac{3D}{\lambda_1 \bar{c}_1} = \frac{6D}{\bar{c}_1 d_p}. \quad [7]$$

This definition of the Knudsen number allows the use of a simple size regime interpolation formula $f(Kn)$, such as the well-known Fuchs and Sutugin expression (4), which adequately approximates the behavior of more rigorous transition regime formulae (5, 6). The Fuchs and Sutugin interpolation function may be expressed

$$f(Kn) = \frac{(4/3)Kn(1 + Kn)}{1 + 1.71Kn + (4/3)Kn^2}. \quad [8]$$

Note the limiting cases,

$$f(Kn) = \begin{cases} 1, & \text{as } Kn \rightarrow \infty \\ & \text{(free molecule limit);} \\ 4Kn/3, & \text{as } Kn \rightarrow 0 \\ & \text{(continuum limit).} \end{cases} \quad [9]$$

Integrating over the size distribution, where $n(d_p)$ is the number density function, provides the total condensation rate R_C.

$$R_C = N_s \frac{\bar{c}_1}{4} \pi \int_{d_k}^{\infty} d_p^2 (S - e^{d_K/d_p}) \times f(6D/\bar{c}_1 d_p) n(d_p) dd_p. \quad [10]$$

It is now convenient to nondimensionalize mass and number concentrations, as well as the rates, by scaling with respect to the properties of the saturated vapor. The vapor concentration will simply be expressed by the saturation ratio S. The aerosol is described by its two dimensionless moments,

$$M = M_p/m_1 N_s \quad [11]$$

$$N = N_p/N_s. \quad [12]$$

A dimensionless surface tension σ^* may be defined by

$$\sigma^* = \frac{2\pi d_1^2 \sigma}{3kT}, \qquad [13]$$

allowing the characteristic Kelvin diameter to be expressed as $d_K = \sigma^* d_1$. A characteristic monomer-monomer collision rate R_β is given by

$$R_\beta = N_s^2 \pi d_1^2 \bar{c}_1/4. \qquad [14]$$

These definitions allow the rate expressions to be written as

$$R_J = R_\beta S^2 \sqrt{\sigma^*/6\pi}\, e^{-\sigma^{*3}/2\ln^2 S} \qquad [15]$$

$$R_C = \frac{R_\beta}{N_s} \int_{d_s}^{\infty} \frac{d_p^2}{d_1^2}(S - e^{\sigma^* d_1/d_p}) \times f(6D/\bar{c}_1 d_p) n(d_p) dd_p. \qquad [16]$$

The equation for R_C may be expressed in terms of the number mean diameter $\bar{d}_p$ or the dimensionless number mean diameter

$$\bar{d}_r = \frac{\bar{d}_p}{d_1} = \left(\frac{M}{N}\right)^{1/3}, \qquad [17]$$

provided that a correction factor α, somewhat less than unity, is introduced to account for the polydispersity,

$$\int_{d_s}^{\infty} \frac{d_p^2}{d_1^2}(S - e^{\sigma^* d_1/d_p}) f(6D/\bar{c}_1 d_p) n(d_p) dd_p = \alpha[\bar{d}_r^2(S - e^{\sigma^*/\bar{d}_r}) f(Kn^*/\bar{d}_r) N_p]. \qquad [18]$$

Defining the molecular Knudsen number as

$$Kn^* = \frac{6D}{\bar{c}_1 d_1}, \qquad [19]$$

the condensation rate may be expressed as

$$R_C = \alpha R_\beta (S - e^{\sigma^*/\bar{d}_r}) \bar{d}_r^2 f(Kn^*/\bar{d}_r) N. \qquad [20]$$

Assuming that the vapor generation rate R_G is constant, the differential equations may be expressed more simply in dimensionless time $\tau = t/\tau_G$ by introducing the time scale $\tau_G = N_s/R_G$, the source regeneration time for the saturated state. Eqs. [1]–[3] now may be written in dimensionless form as

$$\frac{dS}{d\tau} = 1 - g_s J/R^* - C/R^* \qquad [21]$$

$$\frac{dN}{d\tau} = J/R^* \qquad [22]$$

$$\frac{dM}{d\tau} = g_s J/R^* + C/R^*, \qquad [23]$$

where J and C are the dimensionless forms for the rates for nucleation and condensation, respectively, and are given by

$$J = \frac{R_J}{R_\beta} = S^2 \sqrt{\frac{\sigma^*}{6\pi}}\, e^{-\sigma^{*3}/2\ln^2 S} \qquad [24]$$

$$C = \frac{R_C}{R_\beta} = \alpha(S - e^{\sigma^*/\bar{d}_r}) \bar{d}_r^2 f(Kn^*/\bar{d}_r) N. \qquad [25]$$

The dimensionless source rate R^* is defined by

$$R^* = \frac{R_G}{R_\beta}. \qquad [26]$$

It is important to note that classical homogeneous nucleation theory is strictly valid only for R^* values less than approximately unity, since when $R^* > 1$ the vapor concentration will change too rapidly for a steady state cluster profile to develop, and cluster–cluster or cluster–aerosol collisions also may become significant (2).

The set of three simultaneous ordinary differential equations given by Eqs. [21]–[23] are soluble numerically for any given initial conditions. The relevant physical parameters for the system reduce to the following three dimensionless groups: R^*, σ^*, and Kn^*. Of the three, R^* may be varied by changing the source rate, and Kn^* may be varied by changing the pressure of the system, while σ^* is intrinsic to the compound of interest (at a given temperature).

To solve the set of differential equations as posed, it is necessary to select a value of g_s, which has been taken as 200 for the simulations here. A larger than necessary value of g_s does not significantly affect the results, except for R^* approaching unity (or larger), where nucleation ceases to be steady state and the droplet current begins to show a cluster size dependence, and this model begins to break down. An approximation for the polydispersity correction factor α is also required. The simplest is $\alpha = 1$, which corresponds to a monodisperse aerosol distribution, and will be used in these simulations. It can be shown that, for any given N and M, a monodisperse aerosol maximizes the total condensation rate C, and thus $\alpha < 1$ for any polydisperse distribution. But even for very broad aerosol size distributions, or for bimodal distributions with either the mass or number split evenly between the two modes, calculations show $\alpha > 0.6$; only when the bulk of aerosol mass and number occur in different, well-separated modes can α become substantially smaller than one. In such cases where the number and mass concentrations are dominated by two different modes, a pair of differential equations, corresponding to Eqs. [22] and [23], may be set up for each mode. (The homogeneous nucleation term appears only in the smallest size mode.) This bimodal approach will be used when initial aerosol is included.

SIMULATION OF AEROSOL EVOLUTION

The dimensionless model has been used to simulate the evolution of aerosol in an initially aerosol-free system for varying values of the dimensionless parameters R^*, σ^* and Kn^*. As transient behavior has been discussed in an earlier paper (2), the value of the dimensionless number concentration N after the homogeneous nucleation is over is the main item of concern here. Throughout the following discussion, the mention of any aerosol property (e.g., number, mass, source rate, nucleation rate, condensation rate, or surface tension) implicitly refers to the dimensionless form.

As noted above, the model will not be applicable when the vapor source rate R^* is greater than unity, nor when Kn^* is less than unity, since classical nucleation theory requires that the clusters be in steady state with the vapor concentration and that cluster growth occur in the free molecule regime. Calculations have shown that coagulation will be negligible for time scales of the order required to form the aerosol by homogeneous nucleation, provided that R^* is less than unity.

Figures 1 and 2 each show the aerosol number N resulting as a function of source rate R^* for values of surface tension σ^* ranging from 4 to 12. Figure 1 shows results for molecular Knudsen number $Kn^* = 100$, which is typical for atmospheric systems, while Fig. 2 shows results for $Kn^* = 10^5$. In all cases, as the source rate R^* increases, the number of resulting particles increases. This behavior is due to the fact that increasing R^* will drive the saturation ratio higher before increased rates of nucleation and condensation can combine to exceed the increased vapor source rate and thus relieve the supersaturation. And, for any given source rate R^*, a smaller surface tension σ^* will lead to a much faster nucleation rate J without affecting the condensation rate C (for the same saturation ratio S and number N). In general, since particle number is a result of nucleation, while particle mass and vapor depletion primarily result from condensation, it is clear that any change that favors nucleation over condensation will result in a larger number

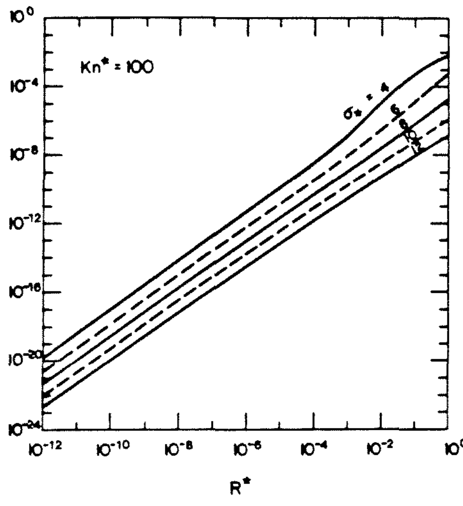


FIG. 1. Predicted dimensionless aerosol number concentration N as a function of dimensionless source rate R^* for various dimensionless surface tensions σ^* with molecular Knudsen number $Kn^* = 100$.

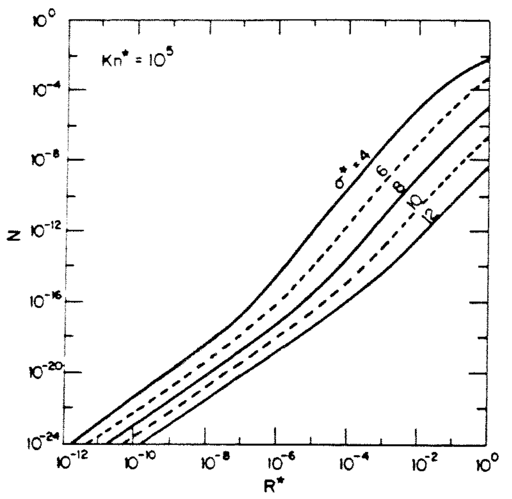


FIG. 2. Predicted dimensionless aerosol number concentration N as a function of dimensionless source rate R^* for various dimensionless surface tensions σ^* with molecular Knudsen number $Kn^* = 10^5$.

of particles. A comparison of the first two figures shows that a larger molecular Knudsen number Kn^*, which will increase the rate of condensation C in the continuum regime, does indeed decrease resulting number N, provided continuum regime growth applies during the burst of nucleation. It is significant to note that for most of Fig. 1 and for the lower number region of Fig. 2, resulting number N shows a dependence close to $R^{*1.4}$, which is evidently characteristic for the case of homogeneous nucleation competing with condensational growth in the continuum regime. Since peak nucleation rates in these simulations typically occur with mass M of order unity, the mean Knudsen number should be unity (transition regime) for number N equal to Kn^{*-3}, which is about where the curves of Figs. 1 and 2 change slope. (The number N at the nucleation peak should be slightly over half of the ultimately resulting number.) For higher values of number N, where the peak nucleation rate J competes with free molecule regime condensational growth, the resulting number N apparently increases approximately as $R^{*2.5}$.

The functional dependence of resulting number N on the dimensionless parameters can be explained by a close examination of the governing equations, combined with a few observations made from the simulations. Regardless of the growth regime of the freshly nucleated aerosol, if the source rate $R^* < 1$, the vast majority of vapor that goes to the aerosol phase condenses onto supercritical nuclei rather than homogeneously nucleates, so condensation dominates over nucleation, i.e., $C \gg g_s J$. (Otherwise, the model would become unacceptably dependent on g_s, which is not observed.) Thus, at the peak rate of nucleation J, where saturation ratio S also reaches its maximum value, and whereupon the number N has achieved over half its ultimate value, the rate of condensation C must equal the source rate R^*, by Eq. [21]. Since the rate of condensation C is proportional to $N^{1/3}M^{2/3}$ in the free molecule regime and proportional to $N^{2/3}M^{1/3}$ in the continuum regime, and assuming the value of aerosol mass M at the peak nucleation time is independent of number N and source rate R^*, we may estimate that the final number N would have a dependence on R^{*3} for free molecule condensation and on $R^{*1.5}$ for continuum condensation. More generally, assuming the Kelvin effect is negligible for the growing aerosol (which is confirmed by the simulations), at the peak nucleation rate,

$$N = \begin{cases} \left(\dfrac{R^*}{\alpha(S-1)}\right)^3 M^{-2} & \text{free molecule;} \\ \left(\dfrac{0.75R^*}{\alpha(S-1)Kn^*}\right)^{1.5} M^{-0.5} & \text{continuum.} \end{cases} \qquad [27]$$

Since simulations reveal that particle mass M at the peak nucleation rate will increase somewhat with larger source rate R^* (or any other change that increases R^*/C for fixed M and N), the power dependence of number N on source rate R^* should be somewhat less than just estimated, and well in line with the power dependences shown by the numerical simulations for two size regimes. Simulations also corroborate the expected dependence of N on Kn^*, as the resulting N goes approximately as $Kn^{*1.4}$ when nucleation competes with continuum condensation, while the resulting N is, of course, independent of Kn^* when nucleation competes with free molecule aerosol growth. Hence simulations with higher values of Kn^* will give curves identical to the free molecule regime portion of Fig. 2, with the steep descent in N with decreasing R^* continuing to lower R^*, until continuum growth finally appears.

Lothe-Pound nucleation theory, which is the primary alternative to classical nucleation theory; typically predicts nucleation rates that are 15 to 20 orders of magnitude higher than classical predictions. Rather than introduce the additional parameters that are needed to characterize a system for Lothe-Pound nucleation, Fig. 3 shows the effect of multiplying the classical nucleation rate J by factors of 1, 10^{10}, and 10^{20}, for the case of surface tension $\sigma^* = 8$ and monomer Knudsen number $Kn^* = 10^5$. Since the rate of homogeneous nucleation is an extremely strong function of the saturation ratio S, while condensation is essentially linear in S, it should not be a great surprise that a substantially faster rate of nucleation (for any given S) would only slightly increase the predicted resulting number of particles, as the burst of nucleation occurs at slightly lower values of the saturation ratio S, where the condensation rate onto each particle is slower so condensation can not quench nucleation until a somewhat larger number of growing particles have been formed. Thus Fig. 3 indicates that the much faster nucleation rates similar to those given by Lothe-Pound theory would increase resulting number concentration by about half an order of magnitude when growth at the peak nucleation rate occurs in the continuum regime, or by about two to three orders of magnitude when it occurs in the free molecule regime. The effect is comparable to a moderate reduction in surface tension, as is seen by comparing Figs. 2 and 3. The qualitative dependence of N on the dimensionless parameters is not changed.

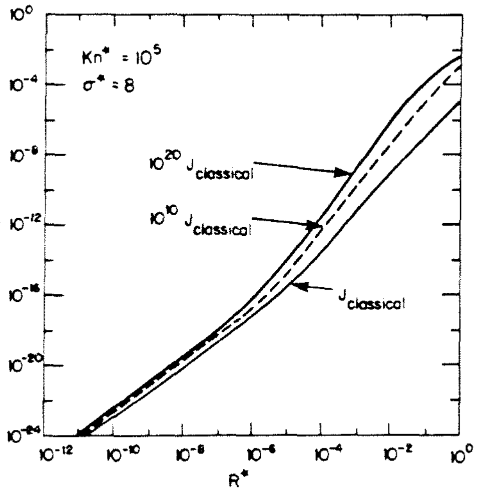


FIG. 3. Predicted dimensionless aerosol number concentration N as a function of dimensionless source rate R^* for nucleation rates of 1, 10^{10}, and 10^{20} times classical, with dimensionless surface tension $\sigma^* = 8$ and molecular Knudsen number $Kn^* = 10^5$.

One expects that a sufficiently small amount of initial aerosol will have a negligible effect on new particle nucleation, while a sufficiently large quantity of initial aerosol will lead to a rapid rate of condensation, preventing the saturation ratio from rising high enough to allow homogeneous nucleation to occur. Figure 4 illustrates the dependence of resulting dimensionless aerosol number on initial dimensionless aerosol number, for cases where the surface tension σ^* equals 8, the molecular Knudsen number Kn^* equals 100, and the initial aerosol is monodisperse with diameter $\bar{d}_r$ equal to 100. Each of the four similar curves for different source rates R^* shows identical behavior. For a small initial number of particles, the final number concentration goes to its limiting value for the case of no initial aerosol (see Fig. 1), which is larger for faster source rates. An effect of the initial number of particles on the ultimate number begins to show only when the initial or seed aerosol number has

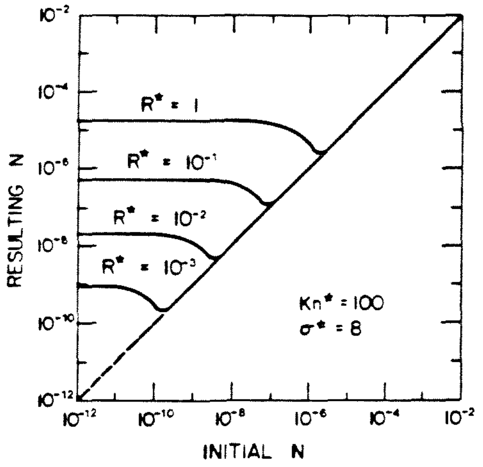


FIG. 4. Predicted dimensionless aerosol number concentration N as a function of initial dimensionless aerosol number concentration for various dimensionless source rates R^* with dimensionless surface tension $\sigma^* = 8$, molecular Knudsen number $Kn^* = 100$, and initial dimensionless aerosol diameter, $\tilde{d}_i = 100$.

risen to at least one-hundredth of this limiting aerosol-free result, at which point increasing the initial number will reduce the resulting number. The resulting aerosol number goes through a minimum corresponding to almost an order of magnitude reduction in final aerosol number over the aerosol-free case, and here almost all the resulting aerosol particles began as initial aerosol, with homogeneous nucleation greatly suppressed. The presence of further initial aerosol entirely suppresses homogeneous nucleation, and the resulting aerosol number simply equals the initial aerosol number. Hence, to minimize the number and maximize the size of particles formed from a condensable vapor, one should begin with a seed aerosol number concentration approximately an order of magnitude less than the number concentration which would have resulted from the absence of initial aerosol under the same conditions. Decreasing the source rate R^* of condensable vapor also leads to significantly fewer but larger particles. If the mean size of the initial aerosol is larger, a somewhat lesser number (but greater mass) is sufficient to suppress nucleation.

CONCLUSIONS

It is possible to predict the number of particles that will be formed by a burst of homogeneous nucleation in a closed system. It is shown that, in the absence of initial aerosol, there are three dimensionless parameters that determine the number of particles which will result: the dimensionless source rate of vapor, the dimensionless surface tension, and an appropriately defined molecular Knudsen number. The resulting number of particles rises more than linearly with increasing dimensionless source rate and with decreasing dimensionless surface tension. The actual functional dependence for the resulting number concentration depends on the regime of condensational growth that competes with and eventually suppresses the homogeneous nucleation. The model presented also allows us to consider systems with initial aerosol, predicting that a seed aerosol concentration equal to about one-tenth the number concentration that would form from an initially aerosol-free system will be sufficient to suppress homogeneous nucleation and will minimize the number of resulting particles.

ACKNOWLEDGMENT

This work was supported by National Science Foundation Grant ATM-8208625.

REFERENCES

1. Warren, D. R., and Seinfeld, J. H., *Aerosol Sci. Technol.* **3**, 135 (1984).
2. Warren, D. R., and Seinfeld, J. H., *Aerosol Sci. Technol.* **4**, 31 (1985).
3. Springer, G. S., *in* "Advances in Heat Transfer" (T. F. Irvine and J. P. Hartnett, Eds.), Vol. 14. Academic Press, New York, 1978.
4. Fuchs, N. A., and Sutugin, A. G., *in* "Topics in Current Aerosol Research" (G. M. Hidy and J. R. Brock, Eds.), Vol. 2, p. 34. Pergamon, Oxford, 1971.
5. Davis, E. J., *Aerosol Sci. Technol.* **2**, 121 (1983).
6. Pesthy, A. J., Flagan, R. C., and Seinfeld, J. H., *J. Colloid Interface Sci.* **91**, 525 (1983).

CHAPTER 5:

HOMOGENEOUS NUCLEATION BY CONTINUOUS MIXING OF HIGH TEMPERATURE VAPOR WITH ROOM TEMPERATURE GAS

Accepted for Publication in

Aerosol Science and Technology

(1986)

HOMOGENEOUS NUCLEATION BY CONTINUOUS MIXING OF HIGH TEMPERATURE VAPOR WITH ROOM TEMPERATURE GAS

Kikuo Okuyama and Yasuo Kousaka
Department of Chemical Engineering
University of Osaka Prefecture
Sakai 591, Japan

Dale R. Warren, Richard C. Flagan and John H. Seinfeld
Department of Chemical Engineering
California Institute of Technology
Pasadena, CA 91125

ABSTRACT

The formation of aerosol particles by homogeneous nucleation in a supersaturated vapor has been studied experimentally and theoretically. In the laboratory, a particle-free gas at room temperature is continuously mixed with a high temperature gas containing dibutylphthalate (DBP) vapor in a new device for the study of aerosol nucleation called a particle size magnifier (PSM). A highly supersaturated vapor is rapidly formed in the mixing zone of the PSM, and the resulting number concentrations of aerosol particles are measured under various temperatures, mixing ratios, and mixing methods. Measured number concentrations are compared with those predicted by the classical and Lothe-Pound nucleation theories. The measured concentrations lie between the predictions of the two theories, and the trends with temperature and saturation ratio are consistent with either nucleation theory, provided vapor depletion is considered.

Introduction

Formation of aerosol particles by homogeneous nucleation commonly occurs in two situations: (1) during the physical cooling of a vapor, or (2) as a result of a gas phase chemical reaction that produces a condensable product. In both cases vapor clusters, growing by successive collisions with the monomer vapor, become large enough to serve as condensation nuclei. Once the clusters exceed a critical size, given by the Kelvin diameter, they tend to grow rapidly by condensation of vapor. Because nucleation rates are exceptionally sensitive to the vapor concentration, laboratory studies naturally tend to use the more easily characterized case (1) system.

In this paper we present a new apparatus for the study of homogeneous nucleation rates under well-defined conditions of temperature, saturation ratio, and residence time. This device, termed a particle size magnifier (PSM), originally developed by Kogan et al. (1960) and modified by Okuyama et al. (1984), has been used previously to detect small condensation nuclei. As a device to measure homogeneous nucleation rates, the PSM has advantages over some alternative techniques, including simplicity of design and operation, effectiveness of mixing, and ability to achieve a wide range of operating conditions. In the previous applications of the PSM, a room-temperature gas containing ultrafine aerosol particles is continuously mixed at a controlled ratio with a high temperature carrier gas saturated with the vapor of a compound (such as dibutylphthalate (DBP) or dioctylsebacate (DOS)) that has a relatively high boiling point. Since ultrafine aerosol particles are grown to stable droplets of around 1 μm diameter, the PSM serves as a practical condensation nuclei counter (CNC) in conjunction with an ultramicroscope or an optical counter. Because vapor pressure varies exponentially with temperature, as the saturated vapor stream is diluted and cooled in the mixing zone, the supersaturations may become large enough to lead to new particle formation by homogeneous nucleation. Thus, the PSM can serve as an apparatus for the study of homogeneous nucleation rates, which is the subject of the present work.

Homogeneous nucleation during the mixing of high-temperature saturated DBP vapor with a low temperature gas has been studied previously by several researchers. In most of these studies (Higuchi and O'Konski, 1960; Amelin et al.,1971; and Rybin et al., 1977), the hot gas, saturated with vapor, issued as a jet into an annular low-temperature gas stream. In one study (Anisimov and Cherevko, 1985), the mixing occurred in a laminar stream rather than in a turbulent jet. Approximate homogeneous nucleation rates evaluated from the measured number concentration of droplets were compared with various homogeneous nucleation theories. In all these studies, accurate characterization of the mixing is essential for estimating nucleation rates, and detailed modeling is required to calculate and to spatially integrate the nucleation rate. The above investigators found substantially higher nucleation rates than those predicted by classical nucleation theory, or, equivalently, found a calculated surface tension somewhat less than the accepted bulk surface tension. Amelin et al. and Rybin et al. showed that the measured DBP nucleation rate exceeded that predicted by the classical theory (using the generally accepted derivation given by Frenkel, 1945, among others) by roughly six orders of magnitude for typical supersaturations and temperatures. Anisimov and Cherevko obtained results that agreed well with Rybin et al., and both groups found DBP nucleation rates that agreed to within an order of magnitude with Barnard's essentially classical nucleation expression (Barnard, 1953), although they admit that such close agreement is fortuitous. In experiments in which nucleation is achieved by mixing hot and cold gas streams, fast and efficient mixing is crucial, as insufficient mixing will cause large spatial fluctuations in the supersaturation. Also, since high concentrations of aerosol result, it may be necessary to consider the depletion of monomer vapor onto the growing droplets when analyzing the observed nucleation rate. In the present experiment, the number concentration of DBP droplets is measured under various gas temperatures, saturation ratios, and mixing methods. The object of the present study is to evaluate the PSM as a device for conducting nucleation experiments and to compare the measured nucleation rates with those predicted by classical and Lothe-Pound theories.

Experimental Apparatus and Method

A schematic diagram of the experimental apparatus is shown in Figure 1. A high purity, particle-free nitrogen gas stream is split into a room temperature diluent gas flow and a high temperature carrier gas flow that is saturated with DBP vapor after passing through the saturator. The two flows are turbulently mixed in the mixing unit, and then held at constant temperature as they flow through the reheater, where the supersaturated vapor is given time to homogeneously nucleate. The resulting aerosol stream then goes to particle measurement instrumentation.

The saturator consists of two columns filled with silica gel that has been impregnated with liquid DBP. The flow rate of carrier nitrogen gas may range from 0.1 to 0.5 l/min, with a corresponding gas residence time varying from 13 sec to 2.6 sec. Temperatures measured by alumel-chromel thermocouples at four positions in the saturator were consistent to within ±0.3 °C. A thermo-controller maintained steady saturator temperature to within ±0.1 °C.

Figure 2 depicts two alternate mixing units that are used to rapidly mix the high-temperature, saturated vapor having temperature T_{sh} with low-temperature diluent gas having temperature T_l. Both mixing units had evolved from earlier work with the PSM, and both seemed to give quite efficient mixing. In mixing unit I, particle-free nitrogen gas saturated with DBP vapor flows horizontally into a tube and meets low-temperature nitrogen gas blown in through eight 0.8 cm diameter holes. In mixing unit II, saturated vapor is forced through a small pipe and becomes an upward jet, mixing with the diluent nitrogen which travels downward through a small tube. The gases are mixed together in the narrow annular gap and flow into the lower pipe. The mixing ratio R_h is expressed as Q_{sh}/Q_m, where Q_{sh} and

Q_m are flow rates of saturated gas and mixed gas, respectively, measured at room temperature. For these experiments, a mixing ratio of either 0.1 or 0.2 was used. The initial vapor temperatures were varied from 105 °C to 125 °C, and the flow rate of room-temperature gas (Q_l) ranged from 0.8 l/min to 2.0 l/min.

Figure 3 shows the reheater section of the PSM, which consists essentially of a wide, temperature-controlled pipe. The DBP vapor and nitrogen mixture flows from the mixing unit to the reheater, which provides the desired residence time t_r for homogeneous nucleation and condensational growth to occur. A thermo-controller is used to maintain the reheater temperature T_r equal to the adiabatic mixing temperature of the vapor-nitrogen stream, $T_m = T_{sh}R_h + T_l(1-R_h)$, in order to maintain constant conditions for nucleation. The volume of the reheater was 190 cm^3, leading to a residence time of 5.7 seconds for a gas flow rate Q_m of 2 l/min. Under steady state conditions, the five temperatures measured by thermocouples shown in Fig. 3 were found to be identical to within 0.5 °C.

When the number concentration of droplets was lower than about 10^3 cm^{-3}, the particles were detected by an optical counter whose minimum detectable diameter is about 0.3 μm. Higher number concentrations were measured in the observation cell using a TV camera with a 25 mW He-Ne gas laser beam to illuminate individual aerosol particles. For either method, particles must grow larger than about 0.3 μm to be detected. In order to detect particles smaller than about 0.3 μm, a highly sensitive TV camera having a minimum detection limit of about 0.07 μm diameter, and a mixing type CNC (Kousaka et al., 1982) having a lower limit of 0.005 μm diameter, were employed.

In starting up the PSM, the two nitrogen flow rates were first set to the desired values, and the temperature of the saturator was increased gradually to the desired temperature. The temperature of the reheater was simultaneously controlled so that it would be at the temperature as determined by the heat and mass balances of both gases. Once steady state conditions were achieved, the number concentration of homogeneously nucleated DBP droplets could be measured.

Figure 4 shows the values of the initial supersaturation ratio S_0 in the mixing zone, which were calculated by heat and mass balances assuming adiabatic conditions (Okuyama et al., 1984). It can be seen that the values of S_0 depend strongly on the temperatures of both gas streams and on the mixing ratio. It is also seen that S_0 attains high values when R_h is between 0.05 and 0.3. Fig. 4 shows that supersaturation ratios ranging from 1 to 1000 may be achieved by changing the various flow and temperature conditions, a range not attainable in most other apparatus. Relatively small values of R_h (0.1 and 0.2) were selected in this experiment to keep the temperature of mixed gas only slightly above room temperature. This created high supersaturations in the reheater and also did not lead to a significant temperature drop between the reheater and the room temperature detector, which could have caused additional, unwanted homogeneous nucleation.

Measured Nucleation Rates

The immediate experimental goal was to measure the number of particles that would form by homogeneous nucleation of DBP in the PSM under a range of initial supersaturations and temperatures. Preliminary experiments demonstrated a clear dependence of particle number on reheater temperature T_r. For given T_m and R_h, a lower reheater temperature leads to a higher number concentration of DBP droplets, as the gas stream saturation ratio rises with cooling. Clearly control of the reheater temperature is essential; since the goal of the experiment is to determine nucleation rate as a function of temperature and saturation ratio, the simplest possible temperature profile, with the reheater temperature set equal to the adiabatic mixing temperature, is the proper way to proceed. The importance of controlling the reheater temperature was also confirmed by the activation experiment of ultrafine particles using the PSM (Kousaka et al., 1985).

Figure 5 compares particle number concentrations of DBP droplets using either mixing units I or II under similar conditions. For both mixing ratios 0.1 and 0.2, the difference in number concentrations for different mixing units was found not to be large. Additionally, the experimental results did not seem to depend on the absolute values of the flow rates of the gas streams, but only on their mixing ratio R_h. For example, Q_{sh}=100 cm^3/min and Q_m=900 cm^3/min gives the same result as Q_{sh}=200 cm^3/min and Q_m=1800 cm^3/min. This indicates that a factor of two in residence time did not make an appreciable difference in the number concentration, although the effect of residence time was not studied in detail.

The mixing time for the production of the highly supersaturated vapor in the mixing unit of the PSM may be estimated as follows. In mixing unit I shown in Fig. 2, the two gas streams are mixed in the pipe from their confluence to the inlet of the reheater. Since the volume of the pipe is estimated to be about 0.07 cm^3, the two gas streams will be mixed within 0.002 sec for a gas flow rate of 2 l/min. In mixing unit II, both gas streams are mixed together in the narrow gap of the mixing unit, having a volume of about 0.06 cm^3. Accordingly, the gas streams will be mixed within a similar time as for mixing unit I.

In this homogeneous nucleation experiment, the supersaturated atmosphere is assumed to be produced instantaneously in the mixing unit, and the new particles are considered to form in the reheater. With the process of condensation of vapor on the vapor clusters or small droplets, heat will be released and the surrounding gas may be warmed. Since the vapor concentration and latent heat of vaporization are relatively small in the case of DBP vapor, the temperature increase of surrounding gas can be neglected, as has been confirmed by numerically solving the basic equations for condensation (Okuyama et al., 1984). This contrasts with the use of water or alcohol vapors in the PSM, where the corresponding temperature increases due to condensation are relatively large, and temperature control of the reheater is much more difficult. Figure 6 indicates the strong dependence of homogeneous nucleation phenomena on the temperature of vapor-nitrogen mixture. Even if the supersaturation ratios are the same, the number concentration of new particles increases with an increase in the temperature of the vapor-nitrogen mixture. This is as expected from nucleation theory, as rising temperature increases the vapor pressure and collision frequency, and, even more importantly, lowers the surface tension, which decreases the activation energy for homogeneous nucleation. (For this figure, a residence time of 1.2 seconds was achieved by using a smaller reheater.)

Simulation Of Aerosol Evolution

We now desire to determine if the data obtained in the PSM may be explained on the basis of homogeneous nucleation theory. In the system an initial supersaturation is produced instantaneously by the mixing process, followed by nucleation and growth of particles. The evolution of the resulting aerosol can be described by the first two moments of the aerosol size distribution, the total aerosol number concentration N_p, and the total aerosol mass concentration M_p, in addition to the number concentration of vapor, N_v. We assume that there are only two physical processes of interest: homogeneous nucleation occurring at rate R_J, and condensation occurring at rate R_C, both of which are expressed in units of number per unit volume per unit time. Mass and number balances for the system yield the following:

$$\frac{\mathrm{d}}{\mathrm{d}t}N_v = -g_s R_J - R_C \qquad [1]$$

$$\frac{\mathrm{d}}{\mathrm{d}t}N_p = R_J \qquad [2]$$

$$\frac{\mathrm{d}}{\mathrm{d}t}M_p = m_1 g_s R_J + m_1 R_C \quad . \qquad [3]$$

This is identical to the approach followed by Warren and Seinfeld (1985), except that the system is now driven by the initial supersaturation ratio rather than by a continuous vapor source term. The somewhat arbitrary diameter at which nucleated clusters are considered to be particles will be denoted by d_s, corresponding to g_s molecules, each of mass m_1. The value selected for g_s is found to have negligible effect on predicted results for cases where the steady state assumption of nucleation holds, so long as g_s slightly exceeds the largest critical number at which significant nucleation can occur, so each freshly nucleated particle continues to grow spontaneously. (Subcritical clusters tend to spontaneously evaporate.)

Expressions for R_J and R_C are available from nucleation theory, which is surveyed by Springer (1978), and by particle mass transfer theory, summarized by Davis (1983). As expressed in a previous paper (Warren and Seinfeld, 1985), the expressions for classical (Becker-Doring-Zeldovich) homogeneous nucleation rate $R_{J_{Cl}}$ and condensational growth using the well-known Fuchs-Sutugin transition regime expression are:

$$R_{J_{Cl}} = R_\beta S^2 \sqrt{\sigma^\star / 6\pi}\, e^{-\sigma^{\star 3}/2 \ln^2 S} \qquad [4]$$

$$R_C = \frac{R_\beta}{N_s} \int_{d_s}^{\infty} \frac{{d_p}^2}{{d_1}^2} \left(S - e^{\sigma^\star d_1 / d_p} \right) f(6D/\bar{c}_1 d_p) n(d_p)\, \mathrm{d}d_p \quad . \qquad [5]$$

The saturation ratio S is the ratio of vapor concentration N_v to the saturated vapor concentration N_s. The monomer diffusivity in the background gas is denoted by D. A dimensionless surface tension $\sigma^\star$ is defined by $\sigma^\star = 2\pi {d_1}^2 \sigma / 3kT$, where σ is the surface tension and d_1 is the monomer diameter (extrapolated from liquid state). A characteristic monomer-monomer collision rate R_β is given by

$$R_\beta = {N_s}^2 \pi {d_1}^2 \bar{c}_1 / 4, \qquad [6]$$

where $\bar{c}_1$ is the mean kinetic velocity of the monomer. The size regime interpolation function $f(Kn)$ is defined to go to unity in the kinetic limit where $Kn \to \infty$. The Knudsen number Kn is defined as

$$Kn = \frac{2\lambda_1}{d_p} \frac{3D}{\lambda_1 \bar{c}_1} = \frac{6D}{\bar{c}_1 d_p} \quad . \qquad [7]$$

This definition of the Knudsen number allows the use of a simple size regime interpolation formula $f(Kn)$, such as the often-used Fuchs-Sutugin expression (Fuchs and Sutugin, 1971), which adequately approximates the behavior of more rigorous transition regime formulae (Davis, 1983 and Pesthy et al., 1983). The Fuchs and Sutugin interpolation function may be expressed

$$f(Kn) = \frac{(4/3)\,Kn\,(1+Kn)}{1+1.71\,Kn+(4/3)\,Kn^2} \quad . \qquad [8]$$

The equation for R_C may be expressed in terms of the number mean diameter $\bar{d}_p$ or the dimensionless number mean diameter

$$\bar{d}_r = \frac{\bar{d}_p}{d_1} = \left(\frac{M}{N}\right)^{1/3} \quad , \qquad [9]$$

provided that a correction factor α, somewhat less than unity, is introduced to account for the polydispersity, where α is defined by

$$\int_{d_*}^{\infty} \frac{{d_p}^2}{{d_1}^2}\left(S - e^{\sigma^* d_1/d_p}\right) f(6D/\bar{c}_1 d_p)\, n(d_p)\, \mathrm{d}d_p = \alpha\left[{\bar{d}_r}^2\left(S - e^{\sigma^*/\bar{d}_r}\right) f\left(Kn^*/\bar{d}_r\right) N_p\right] \quad . \qquad [10]$$

Defining the molecular Knudsen number as $Kn^* = 6D/\bar{c}_1 d_1$, the condensation rate may be expressed as

$$R_C = \alpha\, R_\beta\left(S - e^{\sigma^*/\bar{d}_r}\right) {\bar{d}_r}^2 f(Kn^*/\bar{d}_r)\, N_p/N_s \quad . \qquad [11]$$

It is now convenient to nondimensionalize mass and number concentrations, as well as the rates, by scaling with respect to the properties of the saturated vapor. The vapor concentration will simply be expressed by the saturation ratio S. The aerosol is described by its two dimensionless moments, $M = M_p/m_1 N_s$, and $N = N_p/N_s$.

The differential equations may be expressed more simply in dimensionless time $\tau = t/\tau_C$ by introducing the time scale $\tau_C = N_s/R_\beta$, approximately the time between collisions for a monomer molecule in the saturated vapor. Eqs. [1]–[3] now may be written in dimensionless form as

$$\frac{\mathrm{d}S}{\mathrm{d}\tau} = -g_s J - C \qquad [12]$$

$$\frac{\mathrm{d}N}{\mathrm{d}\tau} = J \qquad [13]$$

$$\frac{\mathrm{d}M}{\mathrm{d}\tau} = g_s J + C \quad , \qquad [14]$$

where J and C are the dimensionless forms for the rates of nucleation and condensation, respectively, and are given by

$$J_{Cl} = \frac{R_{J_{Cl}}}{R_\beta} = S^2 \sqrt{\frac{\sigma^\star}{6\pi}}\, e^{-\sigma^{\star 3}/2 ln^2 S} \qquad [15]$$

$$C = \frac{R_C}{R_\beta} = \alpha \left(S - e^{\sigma^\star/\bar{d}_r}\right) \bar{d}_r^{\,2} f(Kn^\star/\bar{d}_r) N \qquad . \qquad [16]$$

The above 5 equations constitute what shall be referred to as the SNM model for classical nucleation (for the case of no vapor source term). Alternately, the nucleation rate J can be taken from Lothe-Pound theory and be given nondimensionally by

$$J_{LP} = \frac{R_{J_{LP}}}{R_\beta} = \left(\frac{1.1 \times 10^{-5}}{h^6 N_s}\right) \left(\frac{x\, \sigma^\star d_1}{ln S}\right)^{12} (\rho_l\, kT)^3\; x^2 S \sqrt{\frac{\sigma^\star}{6\pi}}\, e^{-\sigma^{\star 3}/2x^2 ln^2 S} \qquad [17]$$

where ρ_l is the liquid density, h is Planck's constant, and the ratio of Lothe-Pound to classical critical diameter, x, is given implicitly by $x = 1 - 4\, ln^2 S/\sigma^{\star 2} x^2$.

The set of three simultaneous ordinary differential equations given by [12-14] are soluble numerically for any given initial conditions. The relevant physical parameters of the system reduce to the following two dimensionless groups: $\sigma^\star$, and $Kn^\star$. The initial saturation ratio S_0 and the residence time t_r also need to be known to carry out the simulation. Additionally, values of 200 and 500 were assumed for g_s, and if the results differed it was concluded that the steady state assumption for the cluster profile did not apply, and that a model using steady state nucleation rates was inapplicable. The polydispersity correction factor α was assumed to be equal to unity, which will slightly overpredict the rate of condensation for an aerosol that is not monodisperse.

Comparison of Measured and Predicted Nucleation Rates

A comparison of measured particle number concentrations with concentrations predicted by homogeneous nucleation theory was made under various assumptions. The physical properties used for the calculations are given in Table 1.

Figure 7 compares the resulting number concentration predicted by classical and Lothe-Pound nucleation theories with experimental results using mixing unit II and a residence time of 5.7 seconds, for mixing ratios of 0.1 and 0.2. The dual sets of data for Lothe-Pound predictions are with and without considering the influence of vapor depletion. (Vapor depletion is negligible if the classical rate of nucleation is assumed.) Neglecting vapor depletion simplifies the calculation of aerosol particle number to $N_p = J\ t_r$, since S and, thus J, is then constant throughout the reheater. If Lothe-Pound nucleation theory is used, vapor depletion must be considered; neglecting it, one would predict more particles formed than molecules of monomer initially present. Fig. 7 clearly shows that classical nucleation theory underpredicts the resulting number concentration (by about eight orders of magnitude) while Lothe-Pound nucleation theory overpredicts the number concentration (by about 12 orders of magnitude). Nevertheless, the trends in the data, of rising N_p with rising S_0, and between the two different mixing ratios, are quite well predicted. It should be noted that the Lothe-Pound rates of nucleation are so high for our experimental conditions that even steady state nucleation does not apply, and results become dependent of the value of g_s. (All figures are for $g_s = 200$.)

Figure 8 compares the trends between experiment and predictions more clearly. Vapor depletion has been neglected, and the nucleation rate is taken as either $10^8 J_{Cl}$ or $10^{-12} J_{LP}$. The predicted nucleation rate trends show a greater increase in N_p with rising S_0 than was observed experimentally. This difference could be presumed to be due to vapor depletion by condensation onto a large number of particles, so that the nucleation rate decreases from its initial value, and the next figure confirms this.

Figure 9 shows predictions using enhanced classical nucleation in the SNM model. The nucleation rate J is assumed to be $10^7 J_{Cl}$, $10^8 J_{Cl}$, and $10^9 J_{Cl}$ for these calculations. In doing so there is rather good agreement with experimental results. The comparison of Fig. 9 with Fig. 8 argues strongly that vapor depletion occurs in the system for number concentrations greater than about 10^3 cm^{-3}. In fact, the SNM simulations show that the saturation ratio did decrease substantially whenever approximately 10^3 cm^{-3} particles or more had been produced during the 5.7 sec residence time.

Figure 10 predicts the number of particles formed in one second as a function of saturation ratio for a series of temperatures, with and without vapor depletion. The nucleation rate expression was taken to be 10^8 times classical, for consistency with the experimental results. Note that vapor depletion significantly reduces the number of particles formed when nucleation rates are greater than about 10^3cm^{-3}sec^{-1}, meaning that the duration of homogeneous nucleation is of order one second or less.

Conclusions

The behavior of the PSM apparatus with supersaturated DBP vapor in the absence of initial aerosol can be explained using a simple model that considers homogeneous nucleation and vapor depletion due to condensational growth. Experimental results were quite consistent either with classical nucleation rate enhanced by a factor of 8 orders of magnitude or with Lothe-Pound nucleation rates decreased by 12 orders of magnitude, which is in agreement with previous experimental studies using geometries more difficult for characterization and interpretation of nucleation rates. For number concentrations greater than about 1000 particles per cm^3 under these experimental conditions, vapor depletion by the growing droplets of aerosol is significant.

Acknowledgment

This work was supported by Japan Grant-in-Aid for Environmental Science 60030039 and National Science Foundation Grant ATM-8503103.

References

Amelin, A.G., Vishnepolskaya, I.V., and Belyakov, M.I. (1971). Experimental investigations of the speed of homogeneous condensation. *J. Aerosol Sci.* 2:93–102.

Anisimov, M.P. and Cherevko, A.G. (1985). Gas-flow diffusion chamber for vapour nucleation studies. Relations between nucleation rate, critical nucleus size and entropy of transition from a metastable into a stable state. *J. Aerosol Sci.* 16:97–107.

Barnard, A.J. (1953). The theory of condensation of supersaturated vapors in the absence of ions. *Proc. Roy. Soc.* 220A:132–141.

Davis, E.J. (1983). Transport phenomena with single aerosol particles. *Aerosol Sci. Tech.* 2:121–144.

Frenkel, Y.I. (1945). *Kinetic Theory of Liquids*, Dover, New York.

Fuchs, N.A., and Sutugin, A.G. (1971). High-dispersed aerosols. In *Topics in Current Aerosol Research*. G.M. Hidy and J.R. Brock, eds., Pergamon, Oxford, vol. 2, p. 34.

Kogan, Y.I. and Burnasheva, A.G. (1960). Growth and measurement of condensation nuclei in a continuous stream. *Phys. Chem. Moscow* 34:2630–2639.

Higuchi, W.I. and O'Konski, C.T. (1960). A test of the Becker-Doering theory of nucleation kinetics. *J. Colloid Sci.* 15:14–49.

Kousaka, Y., Niida, T., Okuyama, K., and Tanaka, H. (1982). Development of a mixing type condensation nucleus counter. *J. Aerosol Sci.* 12:231–240.

Kousaka, Y., Okuyama, K., Niida, T., Hosokawa, T., and Mimura, T. (1985). Activation of ultrafine particles by supersaturation in condensational process. *Particle Characterization*, 2:119–123.

Okuyama, K., Kousaka, Y., and Motouchi, T. (1984). Condensational growth of ultrafine aerosol particles in a new particle size magnifier. *Aerosol Sci. Technol.* 3:353–366.

Pesthy, A.J., Flagan, R.C., and Seinfeld, J.H. (1983). Theory of aerosol formation and growth in laminar flow. *J. Colloid Interface Sci.* 91:525–545.

Rybin, E.N., Pankratova, M.E., and Kogan, Y.I. (1977). Spontaneous nucleation in supersaturated vapours of involatile liquids. *Russ. J. Phys. Chem.* 51:617–619.

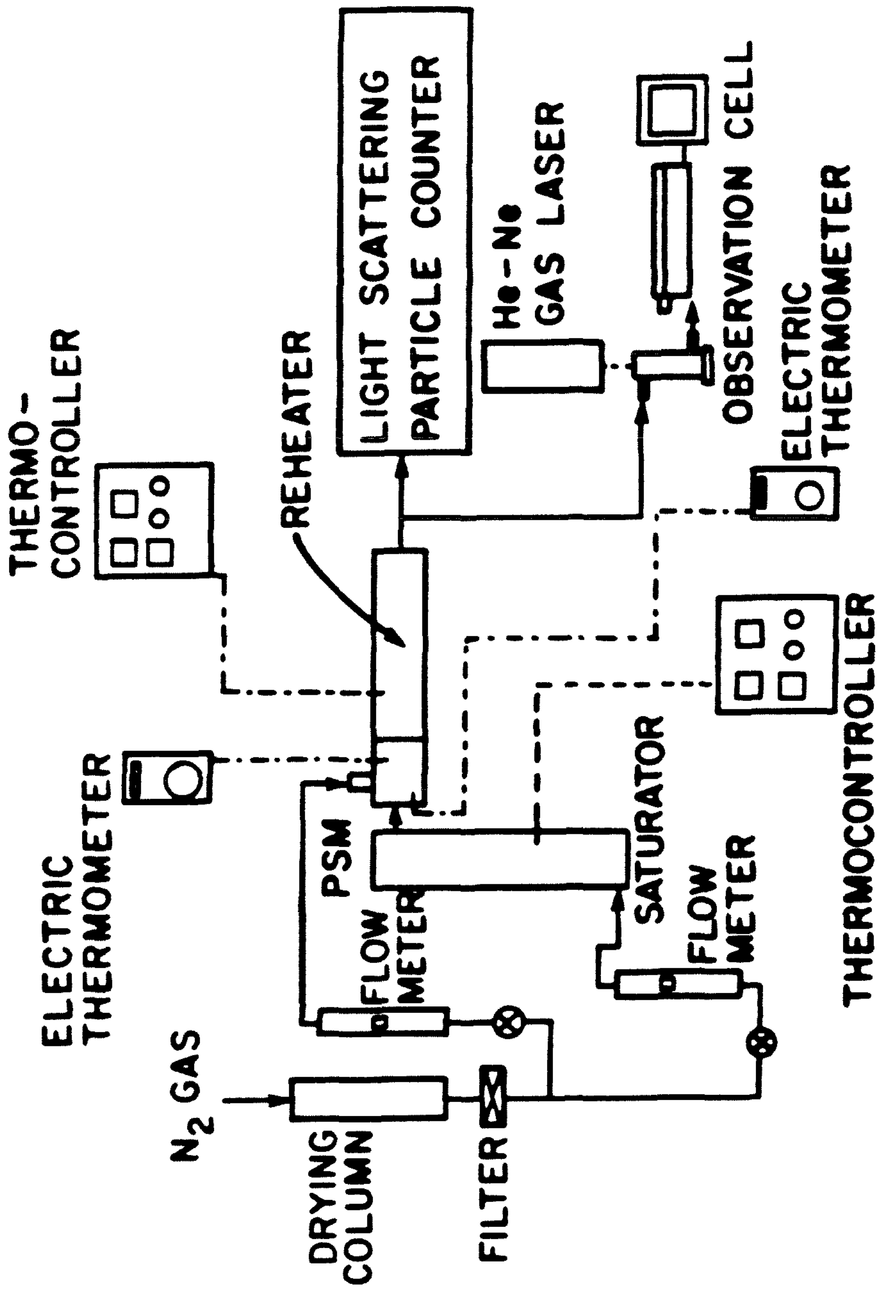
Springer, G.S. (1978). Homogeneous nucleation. In *Advances in Heat Transfer,* T.F. Irvine and J.P. Hartnett, eds., Academic Press, New York, vol. 14., pp. 281–346.

Warren, D.R., and Seinfeld, J.H. (1985). Prediction of aerosol concentrations resulting from a burst of nucleation. *J. Colloid Interface Sci.* 105:136–142.

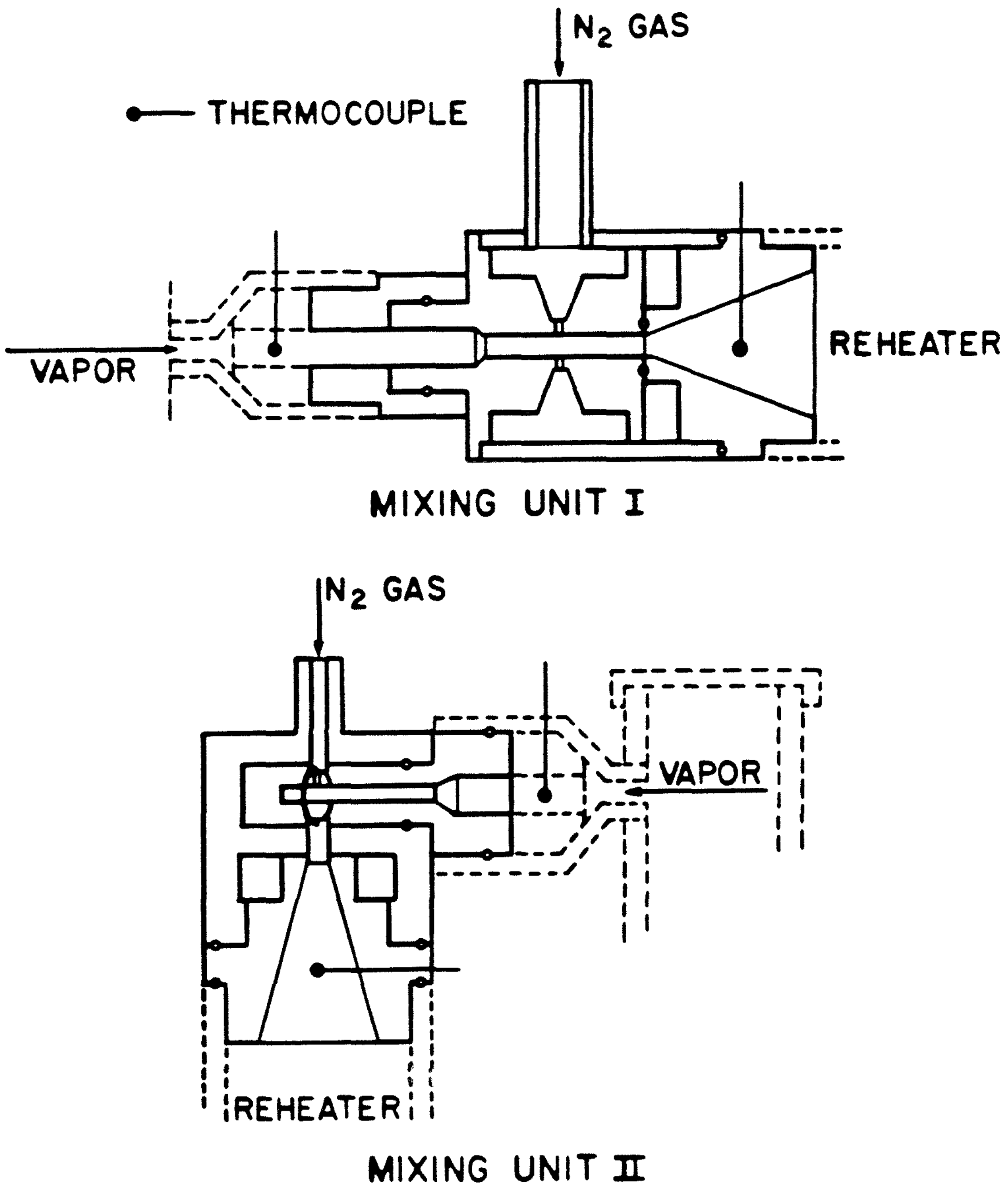
Table 1: Physical Properties of DBP in Air

Molecular weight of DBP, g/mole	$M = 278.35$
Density of DBP, g/cc	$\rho_l = 1.063 - 0.000826\,(T - 273.16)$
Surface tension, dyne/cm	$\sigma = 35.3 - 0.0863\,(T - 273.16)$
Equilibrium vapor pressure, mm Hg	$ln\ p = 16.27 - 5099.0/(T - 109.51)$
Diffusivity†, cm^2/sec	$D = 0.0398\,(T/273.16)^{1.5}$

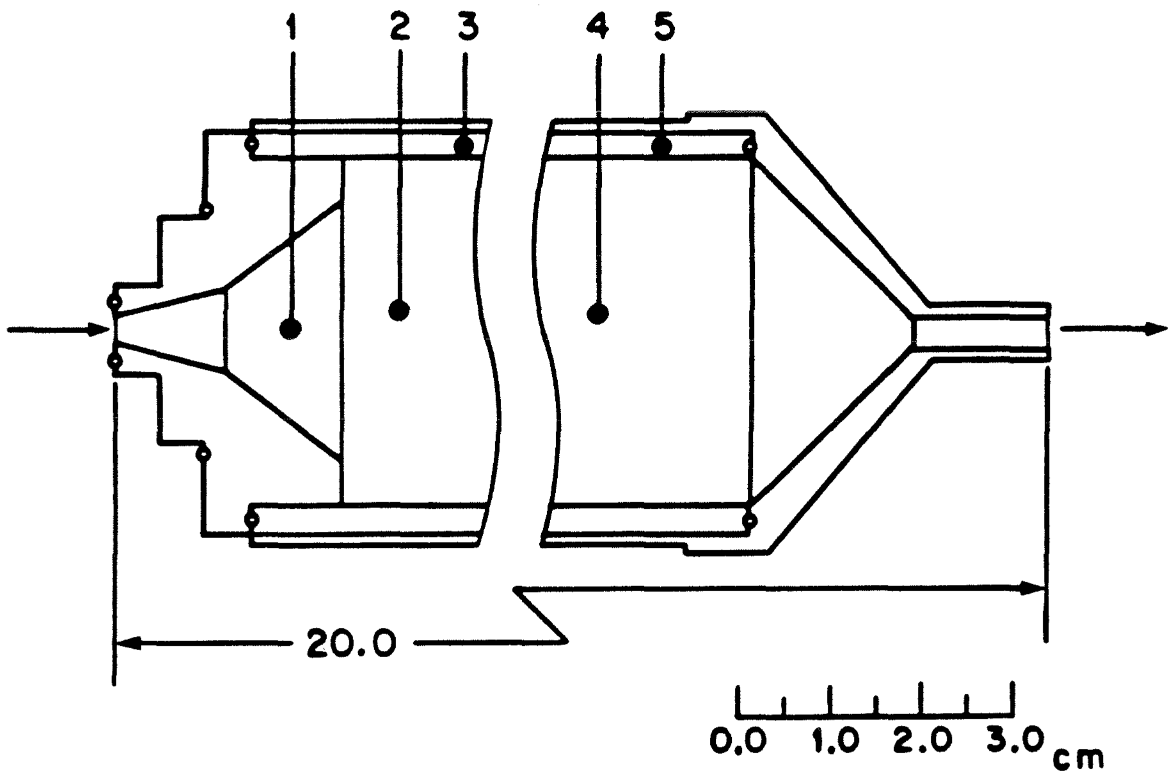
† Linearization of the first-order Chapman-Enskog expression used for calculations.



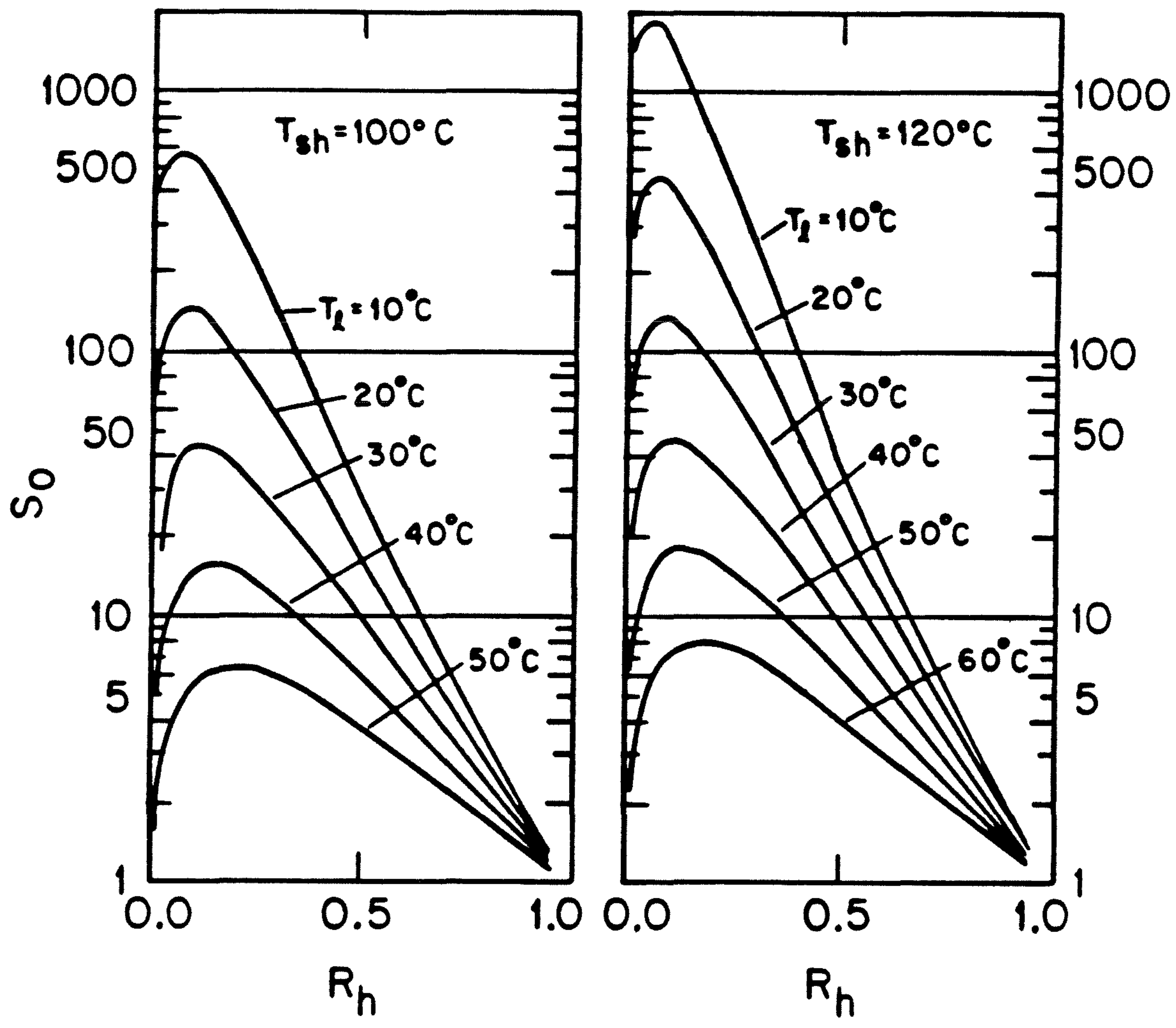
1. Schematic diagram of the experimental apparatus.



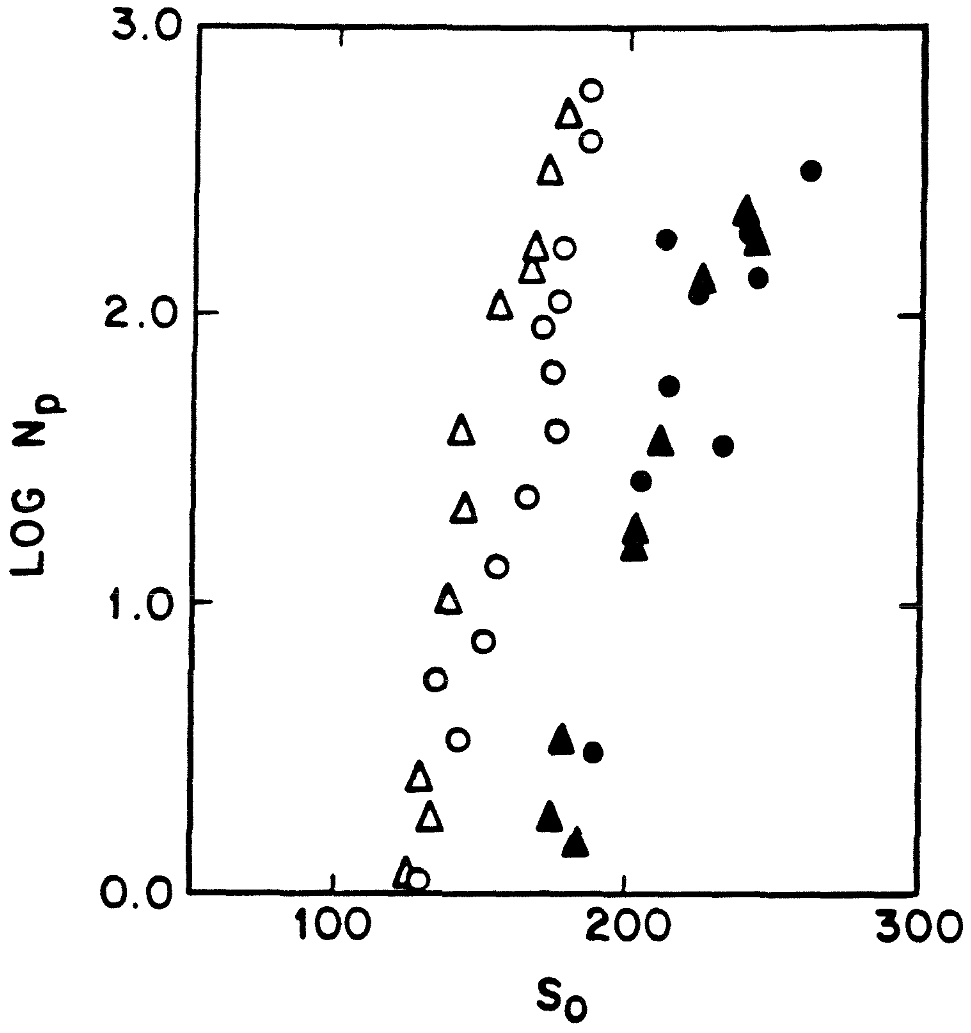
2. The two mixing units used in the Particle Size Magnifier.



3. Diagram of the reheater in which nucleation occurs.

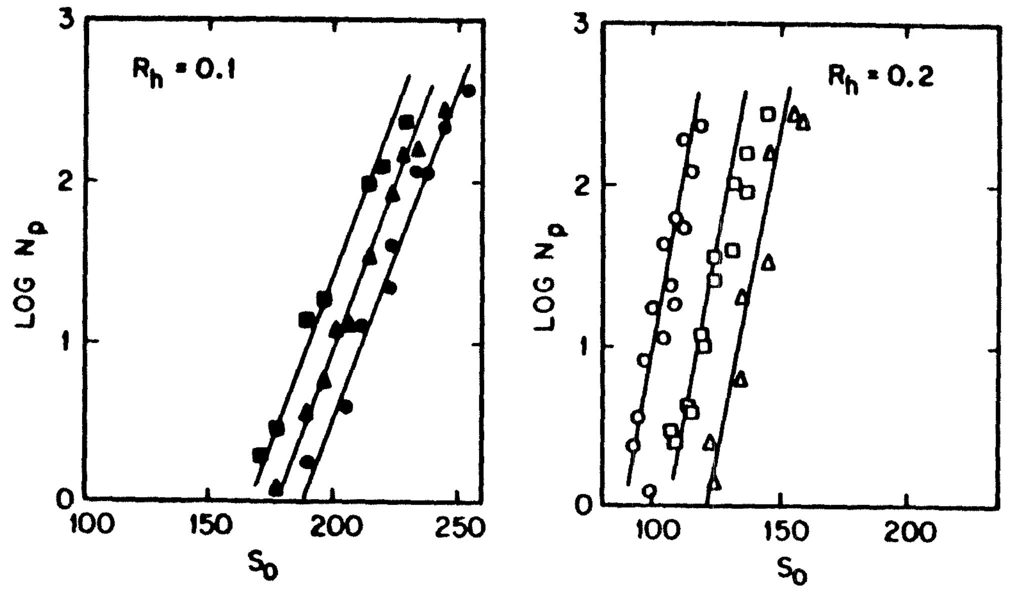


4. Initial supersaturation ratio S_0 under various mixing conditions.



KEY	MIXING	R_h	T_{sh}, °C	T_l, °C
△	I	0.2	105~114	19.8~20.4
▲	I	0.1	102~106	19.1 ~ 19.7
○	II	0.2	107~116	20.1 ~ 20.9
●	II	0.1	103~110	19.2 ~ 20.1

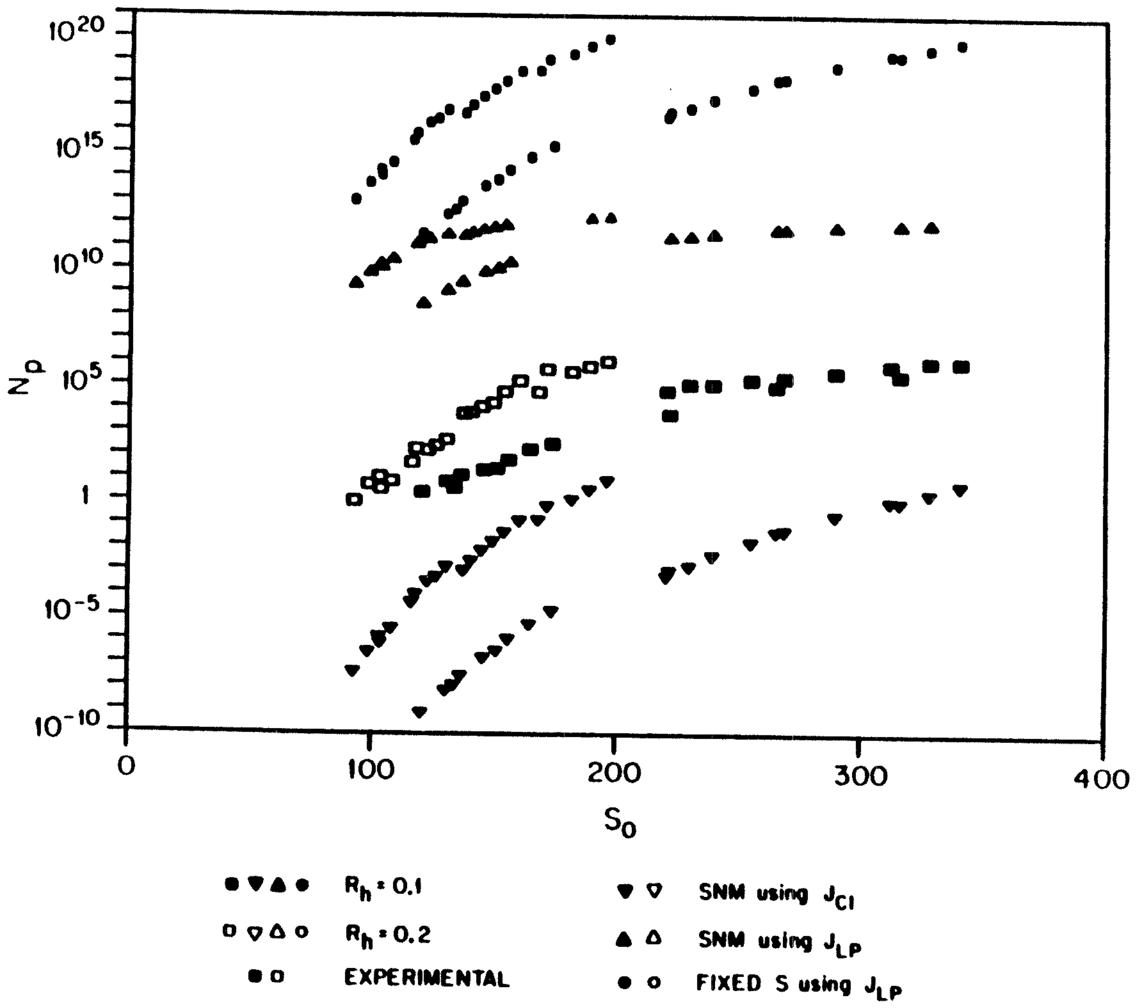
5. Effect of mixing unit on nucleated number concentration (cm^{-3}).



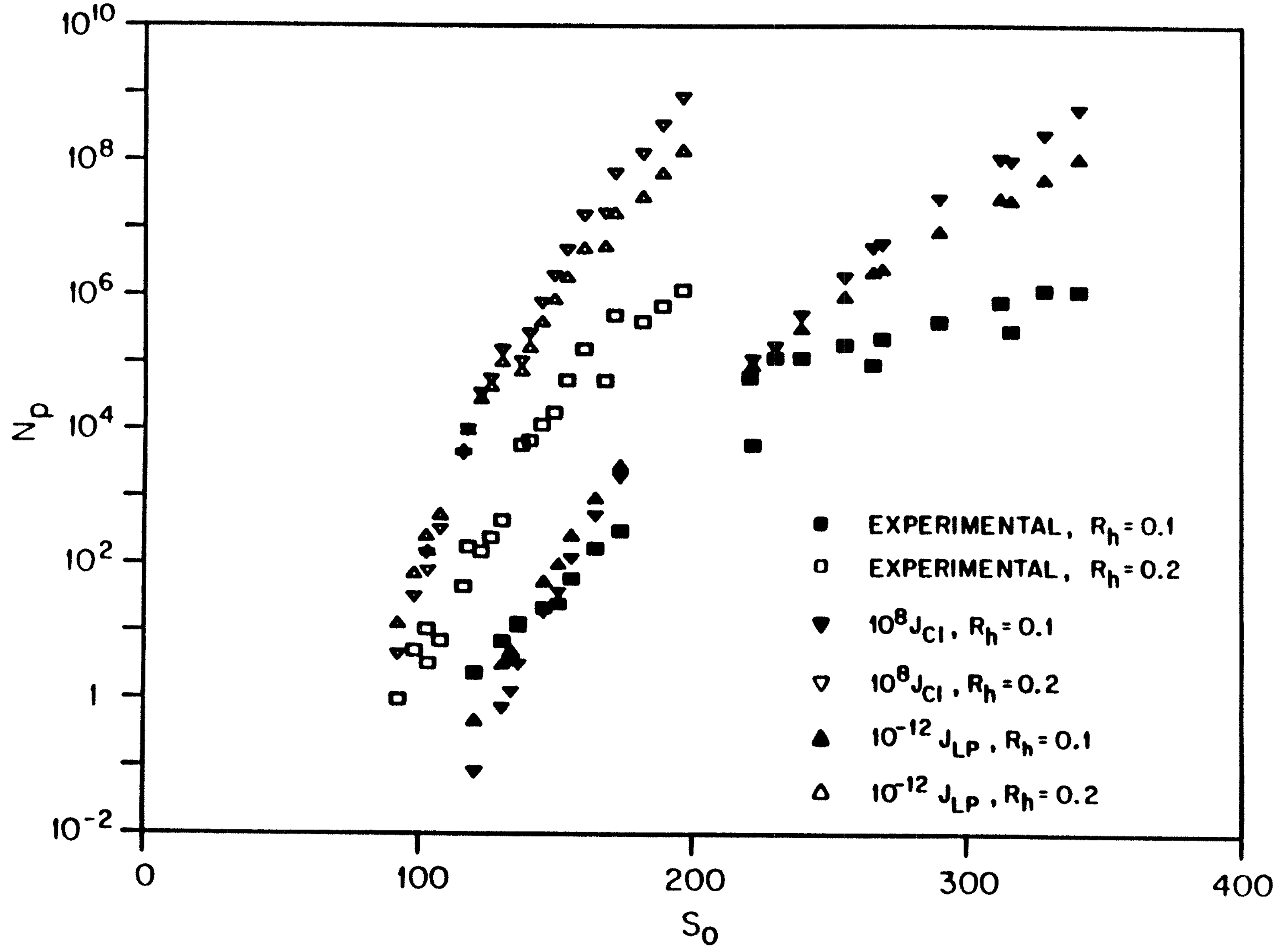
KEY	T_{sh}, °C	T_l, °C	T_m, °C
■	105 ~ 111	21.1 ~ 21.5	29.6 ~ 33.3
▲	105 ~ 109	19.2 ~ 20.2	27.6 ~ 28.6
●	100 ~ 107	17.8 ~ 18.9	26.0 ~ 27.5

KEY	T_{sh}, °C	T_l, °C	T_m, °C
○	108 ~ 115	23.7 ~ 25.0	40.6 ~ 43.0
□	107 ~ 113	22.1 ~ 22.6	39.1 ~ 40.1
△	106 ~ 112	20.6 ~ 20.9	37.7 ~ 39.1

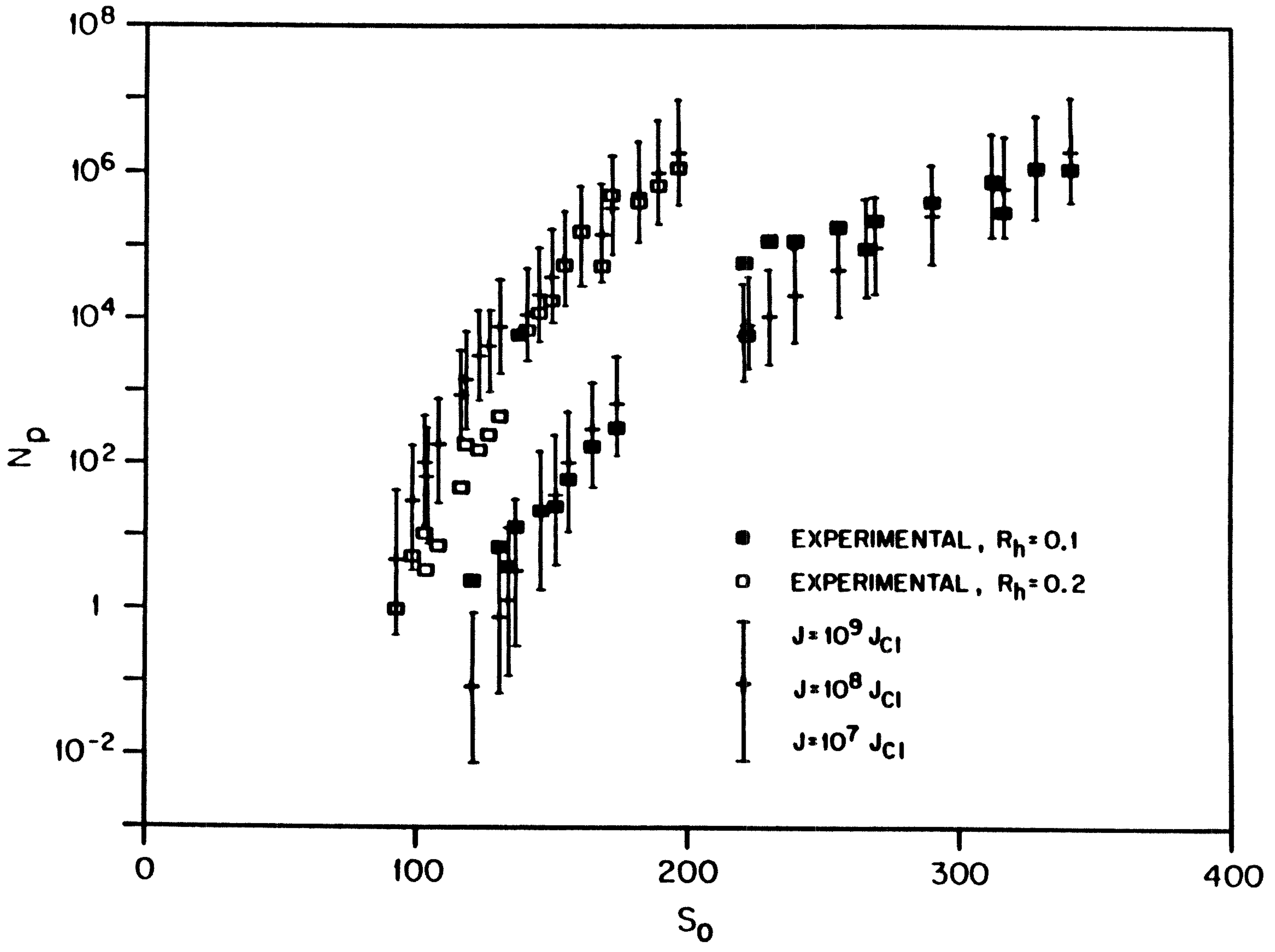
6. Dependence of homogeneous nucleation (cm^{-3}) on the initial saturation ratio and mixing ratio.



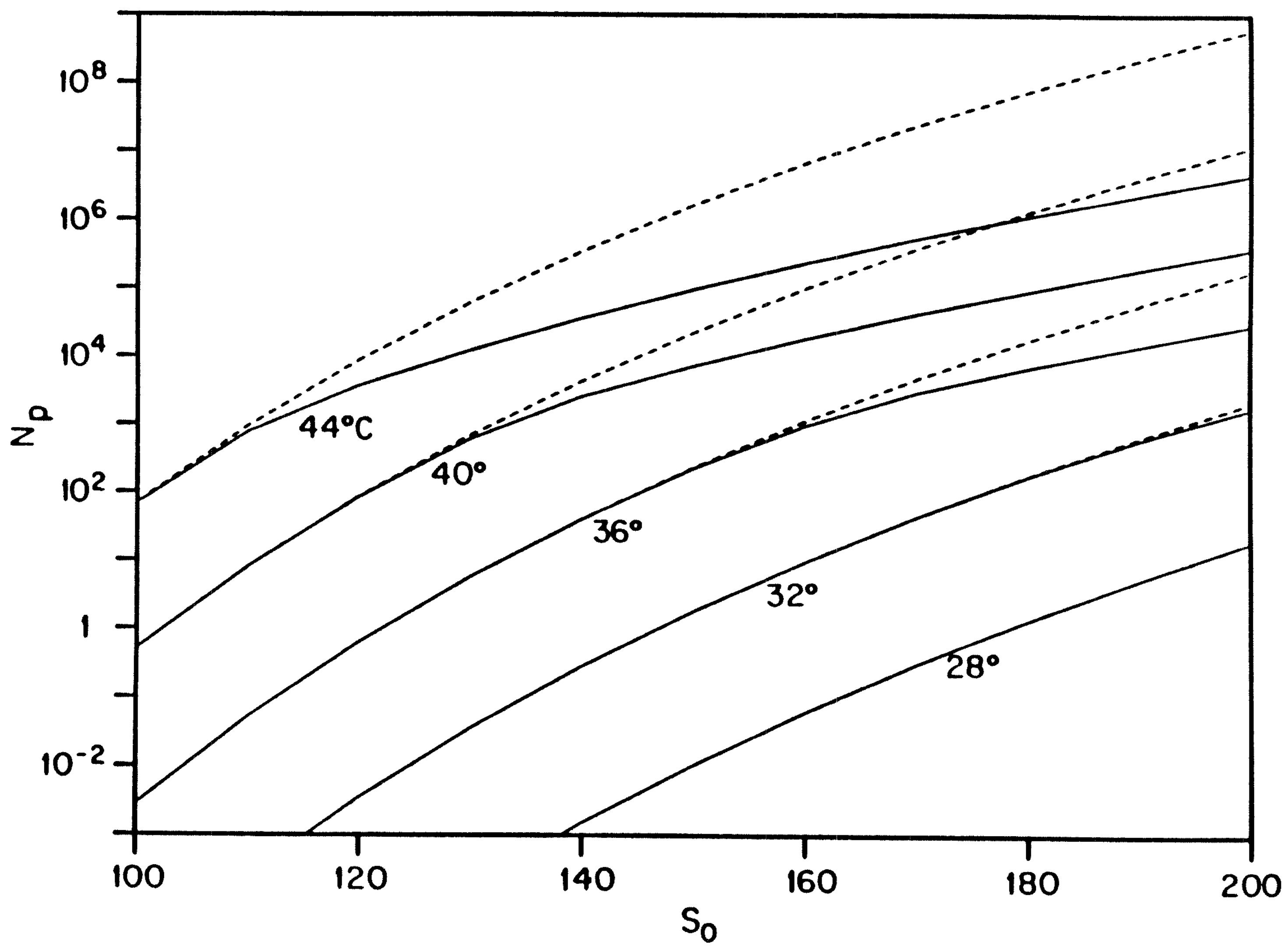
7. Comparision of measured aerosol number concentrations (cm^{-3}) with those predicted by classical and Lothe-Pound nucleation theories.



8. Comparison of measured aerosol number concentrations (cm^{-3}) with those predicted by scaled classical and Lothe-Pound nucleation theories assuming no vapor depletion.



9. Comparison of measured aerosol number concentrations (cm^{-3}) with scaled classical nucleation predictions considering the effect of vapor depletion.



10. Predicted number concentration (cm^{-3}) isotherms by scaled classical nucleation with and without vapor depletion, after one second.

CHAPTER 6:

HOMOGENEOUS NUCLEATION WITH SEED AEROSOL FROM SUPERSATURATIONS FORMED BY RAPID MIXING

Submitted for Publication in

Journal of Colloid and Interface Science

(1986)

HOMOGENEOUS NUCLEATION IN SUPERSATURATED VAPOR CONTAINING FOREIGN SEED AEROSOL

Dale R. Warren†, Kikuo Okuyama◇, Yasuo Kousaka◇,

John H. Seinfeld† and Richard C. Flagan‡

† Department of Chemical Engineering, California Institute of Technology, Pasadena, CA 91125.
◇ Department of Chemical Engineering, University of Osaka Prefecture, Sakai 591, Japan.
‡ Department of Environmental Engineering Science, California Institute of Technology, Pasadena, CA 91125.

ABSTRACT

The formation of aerosol particles by homogeneous nucleation in a supersaturated vapor containing seed aerosol has been studied experimentally and theoretically. In the laboratory, a room temperature gas optionally containing zinc chloride particles is continuously mixed with a high temperature gas saturated with dibutylphthalate (DBP) vapor in a previously discussed device for the study of aerosol nucleation known as a particle size magnifier (PSM). A highly supersaturated vapor is rapidly formed in the mixing zone of the PSM, and gas-to-particle conversion ensues. The vapor may be converted to the aerosol phase by condensation onto the preexisting particles or by homogeneous nucleation to form new particles which then serve as condensation sites themselves. The split between these alternate pathways for gas-to-particle conversion may be deduced from measurements of the resulting aerosol concentrations for different initial supersaturations, seed aerosol concentrations, and seed aerosol sizes. The measured final aerosol concentrations are compared with those predicted by a dynamic model that combines expressions for classical nucleation theory and for steady state particle growth, and agreement is found to within experimental uncertainties. Suppression of homogeneous nucleation by seed aerosol is not predicted to be strong unless seed aerosol number concentrations are larger than the number concentrations which would result from homogeneous nucleation alone.

Introduction

When the partial pressure of a vapor species exceeds its equilibrium vapor pressure, the vapor is said to be supersaturated. Such a state is metastable, as the condensed phase is favored thermodynamically yet the rate of conversion may be very small. The available conversion mechanisms are heterogeneous condensation onto foreign nuclei and homogeneous nucleation onto vapor clusters, which then become effective sites for condensational growth themselves. Supersaturated systems are commonly produced either by the physical cooling of a vapor or by a gas-phase chemical reaction that yields a condensable product. Since the latter method is more difficult to control and the resulting system harder to physically characterize, most laboratory studies including this one employ cooling of a vapor to achieve homogeneous nucleation.

In a previous paper (1), we studied homogeneous nucleation occurring in the absence of seed aerosol under controlled conditions of temperature and initial saturation ratio. The experimental system employed a device referred to as a particle size magnifier (PSM), in which large supersaturations are generated by rapidly mixing a saturated high-temperature vapor stream and room temperature diluent gas, after which nucleation and growth may occur in a relatively large isothermal region downstream. Using dibutylphthalate (DBP) as the condensable vapor with particle-free gas streams, nucleation rates were found to be substantially higher than classical homogeneous nucleation theory predictions, yet substantially lower than Lothe-Pound nucleation predictions; this result was consistent with previous DBP studies (2–5) employing flow configurations such as turbulent jets and laminar streams which are more difficult to model.

In the present paper we shall consider the influence of seed aerosol on homogeneous nucleation, using DBP vapor in the PSM apparatus. It is clear that initial aerosol should deplete the vapor by condensation and reduce the overall amount of nucleation, and that effect may be quantified for various initial supersaturations and seed aerosol concentrations and initial diameters.

Experimental Apparatus and Method

A schematic diagram of the experimental apparatus is shown in Figure 1. The system consists of an evaporation-condensation type aerosol generator, a differential mobility analyzer (DMA), a particle size magnifier (PSM), a light scattering particle counter, and an observation cell. This system previously has been used to study the activation of ultrafine particles as condensation nuclei at sufficiently high supersaturations (6).

A supersaturated vapor is obtained by continuously mixing a room temperature nitrogen gas stream that may contain seed aerosol together with a high temperature nitrogen carrier gas steam that has been saturated with DBP. Mixing occurs in the PSM (7), where the saturated vapor stream is blown through eight 0.8 cm diameter, radially distributed holes into the horizontal tube carrying the room temperature nitrogen stream. A reheater immediately downstream of the mixing region is used to provide sufficient residence time for homogeneous nucleation and condensational growth to occur. A thermocontroller maintains the reheater temperature at the adiabatic mixing temperature of the two nitrogen streams, and initial supersaturations are calculated by heat and mass balances assuming adiabatic conditions.

An ultrafine $ZnCl_2$ aerosol was produced by an evaporation-condensation type aerosol generator (8). The Aerosol from the generator passed through a differential mobility analyzer (DMA), having dimensions as described by Knutson and Whitby (9), which only allowed a nearly monodisperse fraction of $ZnCl_2$ aerosol to enter the PSM with the room temperature gas stream. Depending on the experiment, the mean particle mobility was set by the DMA to correspond to particles either 0.05 μm or 0.10 μm in diameter if singly charged, as the majority will be, especially for the smaller size. The classified particles are charge neutralized by bipolar ions from an Am-241 source, and any charged particles are then removed by a passing the flow between charged plates.

The particles leaving the reheater of the PSM could be counted by a variety of methods, depending on their size and concentration. For particles larger than $0.3\mu m$, an optical counter was used to measure concentrations that were less than 10^3 cm^{-3}, and a TV camera looking at the observation cell under illumination by a 25 mW He-Ne laser was used to measure higher concentrations. For smaller particles, a highly sensitive TV camera could count particles down to 0.07 μm, and a mixing type CNC (8) could count particles down to 0.005 μm.

The experiments were conducted by the following procedure: (1) the flow rates and saturator temperatures were set to their desired values to produce the saturated vapor; (2) the aerosol drawn from the DMA was introduced into the the PSM by diverting the room temperature flow around the filter; (3) the reheater temperature was set to the adiabatic mixing temperature of the two gas streams; (4) the total number concentration of aerosol leaving the reheater was measured once steady state had been achieved; (5) without disturbing the temperature or flow conditions, the seed aerosol was diverted through the filter so that the number concentrations of DBP droplets produced by homogeneous nucleation alone could be measured.

Runs were performed for seed aerosol ranging from 0.050 μm to 0.100 μm in diameter, vapor temperatures from 105 °C to 125 °C, nitrogen flow rates of 0.8 lpm to 2.0 lpm and from 0.2 lpm to 1.0 lpm for the room temperature and high temperature streams, respectively, as measured at room temperature.

Simulation Of Aerosol Evolution

The simulation of number concentrations for the PSM system will be treated by an integral model for nucleation, termed the "SNM model," which was developed previously (1,10). The SNM model derives its name from the dependent variables of which it is comprised through a system of ordinary differential equations; these variables are the vapor saturation ratio S, the total aerosol number concentration N_p, and the total aerosol mass M_p. If there is initial aerosol, the aerosol number and mass are followed separately for the initial aerosol mode and the homogeneously nucleated mode, yielding a system of five simultaneous ODE's which simulates condensational growth, homogeneous nucleation, and, if needed, a source of condensable vapor (and particle deposition, not discussed here). For generality and ease of numerical integration, the entire model is made dimensionless (for isothermal cases) by scaling with respect to the saturated vapor concentration, N_s. The dimensionless time may be inversely proportional either to the intrinsic molecular collision rate of the saturated vapor, or to the regeneration rate of vapor by the source term, as appropriate. (For the PSM system there is no vapor source term; the initial saturation ratio serves as the driving force for particle nucleation and growth.) Using collisional time scaling, the dual mode dimensionless SNM model is given by the following system of equations:

$$\frac{\mathrm{d}S}{\mathrm{d}\tau} = R^\star - g_s J - C_1 - C_2 \qquad [1]$$

$$\frac{\mathrm{d}N_1}{\mathrm{d}\tau} = 0 \qquad [2]$$

$$\frac{\mathrm{d}M_1}{\mathrm{d}\tau} = C_1 \qquad [3]$$

$$\frac{\mathrm{d}N_2}{\mathrm{d}\tau} = J \qquad [4]$$

$$\frac{\mathrm{d}M_2}{\mathrm{d}\tau} = g_s J + C_2 \qquad , \qquad [5]$$

where J and C are the dimensionless rates of nucleation and condensation, respectively. (The seed aerosol mode is denoted by the subscript 1, and the homogeneously nucleated mode by the subscript 2.) Assuming classical homogeneous nucleation theory and the modified Fuchs-Sutugin expression for condensational transport, these rates may be expressed

$$J = S^2 \sqrt{\frac{\sigma^\star}{6\pi}}\, e^{-\sigma^{\star 3}/2 ln^2 S} \qquad [6]$$

$$C_n = \alpha_n \left(S - e^{\sigma^\star/\bar{d}_{rn}} \right) \bar{d}_{rn}^2 f\left(Kn^\star/\bar{d}_{rn} \right) N_n, \qquad n = 1, 2. \qquad [7]$$

These rates of nucleation and condensation have been made dimensionless by dividing the actual rates, in cm^{-3}, by a characteristic monomer-monomer collision rate,

$$R_{11} = \frac{N_s}{\tau_c} = {N_s}^2 \pi {d_1}^2 \frac{\bar{c}_1}{4}, \qquad [8]$$

where $\bar{c}_1$ is the mean kinetic velocity of the monomer and d_1 is the monomer diameter as evaluated from the liquid density. The size regime interpolation function $f(Kn)$ is defined to go to unity in the kinetic limit where $Kn \to \infty$. The dimensionless mean diameter of mode n is evaluated by $\bar{d}_{rn} = (M_n/N_n)^{1/3}$. The polydispersity factors α_n are defined to be the ratio of the condensation rates between the polydisperse aerosol and a monodisperse aerosol with the same total number and mass.

The SNM model, as expressed by equations 1–7, applies immediately to a spatially uniform batch reactor, and will also apply to a one dimensional (plug flow) tubular reactor. The nondimensionalization assumes that the reactor is isothermal in space and time. In dimensionless form, for the case of no seed aerosol, the model indicates that the evolving dimensionless aerosol number concentration will be essentially just a function of the initial saturation ratio or dimensionless source rate that drives the aerosol formation, a dimensionless surface tension, and an effective Knudsen number for the monomer. These three dimensionless physical parameters are defined as follows:

$$\sigma^\star = 2\pi {d_1}^2 \sigma / 3kT \qquad [9]$$

$$Kn^\star = 6D/\bar{c}_1 d_1 \qquad [10]$$

$$R^\star = R_G/R_{11} \qquad [11]$$

where R_G is the source rate of condensable vapor in molecules cm^{-3} sec^{-1}.

From these parameters the SNM model predicts the time evolution of the total number and average size of particles and the vapor saturation ratio in the system.

For cases with pre-existing aerosol, the SNM model also requires the total number of initial particles and their initial mass (or average size), and will then predict the resulting number concentration and size of both aerosol modes. Mathematically, the SNM model may be viewed as a functional relationship,

$$(S, N_1, M_1, N_2, M_2) = g(\tau, R^\star, \sigma^\star, Kn^\star, S_0, N_0, M_0; \alpha_1, \alpha_2, g_s, E_J) \quad , \qquad [12]$$

The system state is a function of time τ, three fundamental physical parameters ($R^\star$, $\sigma^\star$, and $Kn^\star$), the initial conditions (at $\tau = 0$, $S = S_0$, $N_1 = N_0$, $M_1 = M_0$, $N_2 = 0$, $M_2 = 0$), and four subsidiary inputs: the effect of polydispersity of the condensation rates of primary and secondary particles, α_1 and α_2, respectively; the assumed supercritical cluster number g_s at which homogeneously nucleated particles emerge from vapor and join the secondary aerosol mode; and a "nucleation enhancement factor" E_J expressed as the ratio (for any given S and $\sigma^\star$) between J as implemented in the calculations and the classical nucleation expression for J as presented above. In the standard implementation of the model, these subsidiary inputs are treated as follows: α_1 and α_2 are set to unity, so both modes of the aerosol are treated as if they were monodisperse; g_s is set equal to 500; and the nucleation enhancement factor E_J is set to unity if possible, or else to a constant value for a given compound.

Because of the simple nature of the SNM model, there are several questions that must be addressed before it can be considered as an approximate description of a real system, namely (1) the accuracy of describing the aerosol by two modes, (2) the neglect of other physical processes such as deposition and coagulation, (3) the assumption of spatial uniformity, (4) the assumption of steady state nucleation and condensation rate expressions, and (5) the uncertainty of homogeneous nucleation rates.

Clearly the SNM model forfeits information about particle size by treating the number distribution as a pair of delta functions. This simplistic treatment will bias the condensation rate upwards (a monodisperse distribution maximizes total area or total length for any given number and mass of particles, which will maximize the condensation rate unless the Kelvin diameter is close to or greater than the modal diameter), but not greatly (as found by calculating α_1 for a modestly polydisperse system of the sort we are attempting to represent). If subunity values of α_1 and α_2 are chosen, corresponding to true degree of polydispersity at some given time, the condensation rate might be accurate at the given time, but the SNM model is unable to update α_1 and α_2 as the actual size distribution changes in width with condensational growth or nucleation.) Perturbing α_1 or α_2 downward leads to a roughly proportional increase in the number of particles nucleated (assuming the perturbed mode dominates total condensation). So if we assume monodispersity, it will underestimate the number of particles which nucleate in about direct proportion to the polydispersity of the aerosol, but the resulting error in final number concentration is likely to be of order 10% or less for a system that is not highly polydisperse. (The secondary aerosol formed by homogeneous nucleation will not be very polydisperse.)

The SNM model makes certain implicit assumptions about the physical processes occurring. It assumes that coagulation is negligible relative to the time scale on which the nucleation event occurs, which is generally true. It assumes that steady state expressions for nucleation and condensation are appropriate, which may be ascertained by estimating the system time constants. Steady state nucleation applies when the critical cluster concentration is in steady state with a vastly larger instantaneous monomer concentration, as occurs when the dimensionless source rate is less than approximately unity.

The effect of spatial concentration gradients around growing particles is neglected by the SNM model. Comparison of the SNM model with a steady state cell model incorporating spatial gradients has shown negligible differences in aerosol number achieved for both the initial supersaturation and constant source rate systems investigated. Thus it appears that the cell size is sufficiently large during a burst of nucleation so that the vast majority of the vapor has a saturation ratio and nucleation rate very close to the spatial average values (11). In the Appendix, arguments are presented which show that for all dilute systems and even most systems with high mass loadings, the cell model reduces to the SNM model to a precision considerably higher than the nucleation rate can be predicted.

The SNM model (as well as the cell model) assumes that steady state nucleation expressions apply, i.e., that clusters of critical size (and even larger up to g_s) are in a steady state relationship with the instantaneous monomer concentration. If the saturation ratio is changing too quickly, the classical homogeneous nucleation rate expression will not be valid, as a significant time lag will exist between changes in saturation ratio and changes in cluster concentrations. Thus the characteristic time for change in saturation ratio $(d\ln S/dt)^{-1}$ must be longer than the characteristic time lag for steady state cluster concentrations. Classical nucleation theory also fails if the saturation ratio is so large or surface tension so small that the activation energy barrier at the critical size is no longer substantial; then the assumed steady state cluster profile will not apply because of cluster-cluster collisions, and the monomer will rapidly produce a great number of very small supercritical particles which will coagulate before the aerosol reaches measurable size. Classical homogeneous nucleation theory only applies when the monomer number concentration and surface area overwhelm that of the clusters and of the aerosol; then the rate of nucleation (i.e., of production of supercritical clusters) is dominated by the rate at which monomers collide with critical clusters and not appreciably augmented by the collisions of two subcritical clusters to form supercritical clusters. Cluster-cluster collisions should not significantly influence the nucleation rate when approximately $\ln S < 0.4\sigma^\star$ and $\ln S < \sigma^\star - 2$, as then the activation energy barrier will be significant and the monomer concentration will dominate over the total concentration of all clusters; these conditions apply to our experiments and simulations. When steady state nucleation does apply, the exact value of g_s will have negligible effect on the behavior of the SNM model, provided $M_2/N_2 \gg g_s \gg g_c$.

Uncertainty in the homogeneous nucleation expression can lead to appreciable uncertainty on predictions of final number. Since the Lothe-Pound and classical theories of homogeneous nucleation rate typically differ by 15 orders of magnitude in nucleation rate for a given S, this would seem to be a very serious problem. Systems with a vapor source have been shown to be relatively insensitive to the nucleation expression, while systems like the PSM which have a fixed maximum S are quite sensitive to the nucleation expression, as will be shown later.

Number Concentrations Generated in the Particle Size Magnifier (PSM)

Four sets of experiments were performed using DBP and a zinc chloride initial aerosol in the PSM, as summarized in Table I. Those with a mixing ratio of 0.2 (sets A and B) resulted in a PSM temperature in the vicinity of 43 °C, while those with a mixing ratio of 0.1 (sets C and D) led to a mixed temperature around 33 °C. Within each set of experiments, the initial aerosol concentration and size were held constant, while variations in the temperatures of the DBP-saturated and diluent nitrogen streams led to different initial saturation ratios and slightly different temperatures in the PSM. The resulting aerosol number concentrations were measured with and without introducing initial particles, thus providing a matched pair of aerosol number concentrations at each fixed experimental condition.

Figure 2 shows the measured number concentrations generated in the PSM as a function of initial saturation ratio for all 27 experiments in the absence of initial aerosol. The SNM model predictions are plotted also, both as corresponding points based on the experimental temperature, and as continuous lines corresponding to 33 °C and 43 °C (of which any experiment was within ± 1.6 °C). Since classical homogeneous nucleation theory predicts far too slow a rate of nucleation to explain the measured number concentrations, the classical homogeneous nucleation rate was enhanced by a factor of 10^7 in order to obtain agreement between simulations and experiments. Table II presents the predictions of the SNM model using the 10^7 value for the nucleation enhancement factor E_J. The ratio of the predictions using $E_J = 10^7$ to the measured values of N_{J_0} had a mean of 1.10 with a standard deviation of 0.39, and ranged from 0.58 to 2.29 in value. For the earlier set of DBP nucleation experiments in the PSM (1), an enhancement factor of approximately $E_J = 10^8$ over classical nucleation theory was required to obtain agreement between predictions and observations. If the 10^8 enhancement factor were used to simulate the present experimental data, N_{J_0} would be overpredicted by an average factor of 5.22 (standard deviation of 2.00, range 2.63 to 11.46).

When nucleation is driven by a high initial saturation ratio, as in the PSM, the resulting number concentration of nucleated particles is fairly sensitive to the nucleation rate expression (the predicted value of N_{J_0} shows about a two-thirds power dependence on the nucleation enhancement factor in these simulations); this is in marked contrast to a nucleating system driven by a steady monomer source rate, as noted in a previous paper (10). When the monomer is being generated by a continuous source, the saturation ratio builds up coincident with increasing nucleation and condensation until the combined rates of nucleation and condensation deplete the vapor. In that situation large differences in the dependence of J on S merely lead to slightly different maximal S values, which do not greatly affect the rates of nucleation or condensation occurring at the peak S. In the PSM, on the other hand, the maximum S is fixed at the initial value S_0 and the natural balancing of nucleation and condensation rates that occurs in the steady monomer source case does not take place. Consequently, the system becomes much more sensitive to the assumed nucleation rate function.

The very sensitive dependence of resulting number concentration N_{J_0} on the initial saturation ratio and temperature (because of the exponential vapor pressure dependence) makes it extremely difficult to predict N_{J_0} precisely (i.e., to better than about 35%, even after optimizing the enhancement factor E_J) given measured values of S_0 and T. Since the SNM model predicts nucleated number concentrations N_{J_0} that may vary by up to a factor of two from the experimental values, a direct comparison between measured and predicted values of N_f resulting with initial aerosol is not very useful; the suppression effect on homogeneous nucleation due to the initial aerosol would be lost in the uncertainty of how many particles would nucleate without initial aerosol. Fortunately, since the experimental temperature and initial saturation ratio were kept identical (in both the absence and presence of initial aerosol), the effect of initial aerosol on nucleation may be found by scaling each aerosol number concentration to that which resulted from no initial particles,

N_{J_0}. Once scaled to N_{J_0}, the behavior of the system is much less sensitive to S_0 and T, and the degree of suppression of nucleation due to seed aerosol is revealed. Hence we shall focus on the relative initial number concentration, N_i/N_{J_0}, and the relative resulting number concentration, N_f/N_{J_0}, as we assess the effect of initial aerosol on homogeneous nucleation in the PSM system.

Figure 3 shows the relative total number concentration, N_f/N_{J_0}, as a function of relative initial aerosol concentration, N_i/N_{J_0}, for all experiments. Data would adhere to the diagonal line if $N_f/N_{J_0} = N_i/N_{J_0}$, namely that the final number of particles is equal precisely to the initial number of seed particles, that is no new particles formed by nucleation. With any new particle formation by nucleation, $N_f/N_{J_0} > N_i/N_{J_0}$, and the measured values are expected to lie above the diagonal line. On the other hand, the upper curve represents the case in which the number of particles formed by nucleation is uninfluenced by the initial seed aerosol. In that case the final number concentration of particles is simply the sum of that predicted to be formed by nucleation in the absence of seed aerosol and the initial number concentration of seed particles. All results are expected to lie between the lower diagonal line representing no new particle formation by nucleation and the upper curve representing no influence on nucleation by the initial aerosol. It is expected that all experimental data and all simulations should lie between these two limits, representing some degree of suppression of homogeneous nucleation by the initial aerosol. At the left hand side of the figure, where initial aerosol concentrations are low, the final number concentration is essentially identical to that generated by homogeneous nucleation alone. As the relative initial aerosol concentration approaches unity, the relative final aerosol concentration is, for the conditions of the experiments, predicted to be only slightly above unity, indicating partial suppression of nucleation. Here each initial aerosol particle, depending on size, is roughly as effective at depleting the vapor concentration as a homogeneously nucleated particle. (A pre-existing particle of the same size as the growing condensed droplets will remove just as much mass from the vapor phase, except for the fact that its initial mass was not taken from the vapor phase.) As the relative initial aerosol number becomes large, more suppression of nucleation should occur, although nucleation will never be entirely suppressed according to a steady state nucleation model. Unfortunately the difference between the initial and final number concentrations is very difficult to measure in this region, and once $N_i/N_{J_0} > 4$ the measured values of N_f could not be distinguished from either N_i or $N_i + N_{J_0}$ because of the bounds of experimental error.

Figure 4 presents the results of the experiments and simulations in a manner that further elucidates the effect of the seed aerosol on the ultimate number of particles formed. We define the relative nucleated number concentration as $(N_f - N_i)/N_{J_0}$, which is just N_J/N_{J_0}. Any value of this ratio less than unity indicates suppression of nucleation due to the initial aerosol. For $N_i/N_{J_0} \ll 1, N_J/N_{J_0} = 1 - N_i/N_{J_0}$ approximately applies to both the experimental and simulation results. At $N_i/N_{J_0} = 1$, N_J/N_{J_0} is barely over 0.5 according to the SNM simulations, but closer to 0.7 by the majority of the experimental measurements in that region. As N_i/N_{J_0} is increased substantially above one, one expects and the SNM simulations show decreasing nucleation. The fact that the measurements seem to indicate full nucleation is this region must be attributed to the experimental difficulties in measuring small differences between large numbers. The bottom half of Fig. 4 shows the ratio of the final number N_f to the sum of the initial number N_i and the number N_{J_0} produced by nucleation alone as a function of the relative initial number concentration, N_i/N_{J_0}. Any value of $N_f/(N_{J_0} + N_i)$ less than unity indicates suppression of total number due to initial aerosol. A change in total number of five to ten percent represents the present detection limit of the PSM apparatus. Suppression of total number is negligible for N_i/N_{J_0} much less than unity, while for N_i/N_{J_0} of about unity the resulting aerosol is predicted to be about 23% less and observed to be 12% less than the sum of N_i and N_{J_0}. Of the fifteen experimental values having relative initial number concentrations between 0.5 and 3., five are very close to the simulations while ten lie noticeably above the simulations. Hence the experiments indicate about half the peak suppression of resulting number concentrations due to initial aerosol as the SNM simulations predict. Experiments and simulations agree that the total number resulting within the PSM is reduced by at most one-quarter from the sum of the initial aerosol number plus the number of particles that would have been nucleated from the vapor phase had no initial aerosol been present. In four cases the measured final number concentration was slightly larger (by 2 to 9%) than $N_i + N_{J_0}$, which demonstrates the uncertainty of the measurements.

In summary, both the experimental data and the SNM model predict that the resulting number concentration N_f will be at most only slightly less than the sum of the initial aerosol number N_i and the number of particles N_{J_0} formed by homogeneous nucleation alone. From Fig. 3 and 4 is it clear that the interaction between initial aerosol and homogeneous nucleation is not great in the experiments performed in the PSM system. To understand the dynamics of nucleation and condensation better in the system, and search for conditions where a greater suppression of total aerosol number would result, SNM simulations were performed for an initially supersaturated DBP system over a range of initial particle concentrations, initial particle sizes, initial saturation ratios, and temperatures. The typical simulation chosen for comparison had a temperature of 40 °C, and initial saturation ratio of 250, and an initial aerosol diameter of 0.1 μm. In the absence of seed aerosol, simulations showed such a system would yield 1.33×10^6 particles cm^{-3}, so all parameters of the simulation were within the experimentally observed ranges.

Figure 5 shows the predicted suppression of nucleation of total number as a function of relative initial number for initial particle diameters of 0.01 μm, 0.1 μm, and 1.0 μm. The condensing species is assumed to be DBP at 40 °C and with an initial saturation ratio of 250. As would be expected, larger seed particles at the same number concentration caused more suppression of nucleation, and the maximum percentage suppression in total number occurred at a smaller relative initial number. For a seed diameter of 0.01 μm, the maximum number suppression was 19%; for a 0.1 μm seed, the maximum suppression in number was 24%; and for a 1.0 μm seed, up to a 55% reduction in total number is predicted to be achieved.

Figure 6 shows the suppression of nucleation and suppression of total number as a function of relative initial number for initial saturation ratios of 200, 250, and 300, for DBP at 40 °C, and an initial aerosol of 0.10 μm in diameter. As the initial saturation ratio increases, increased suppression of nucleation occurs for the same relative (but greater absolute) number of initial aerosol (because N_{J_0} is itself increasing as S_0 is increased), and the peak reduction in resulting number occurs at lesser relative initial aerosol number. At $S_0 = 200$, the maximum total number reduction is predicted to be 23%; at $S_0 = 250$, 24%; and at $S_0 = 300$, 30%. Note that the nucleated aerosol is predicted to grow substantially larger than the 0.1 μm initial diameter before the burst of nucleation is over. The increasing suppression of total number and of nucleated number with rising S_0 is attributed to the decrease of the activation energy for nucleation with rising S_0, to be discussed later.

The suppression of nucleation and the of total number as a function of relative initial number for a supersaturated DBP system at temperatures of 30, 40, and 50 °C is shown in Figure 7. The initial saturation ratio was fixed at 300 and the seed particle diameter was fixed at 0.1 μm. At 30 °C, the maximum total number suppression was 23%; at 40 °C, 24%, and at 50 °C, 47%. The increasing suppression of total number and of nucleation with increasing temperature is a result of the decreasing dimensionless surface tension, which decreases the nucleation activation energy and hence extends the duration of nucleation over a longer time and broader range of saturation ratio, giving any initial aerosol a longer opportunity to make its presence felt.

Figure 8 compares the suppression of nucleation and of total number as a function of relative initial number concentration between a system such as the PSM with an initial supersaturation and one having a steady source rate of condensable monomer. The results for the aforementioned typical initial conditions of the PSM having an initial saturation ratio of 250 are plotted alongside those of a system with no initial vapor but dimensionless source rates of 10^{-4} and 1. In all cases DBP at 40 °C is taken as the condensing species. Note the drastically sharper and stronger suppression of nucleation in the source-rate systems. The up to 80% reduction in total number for the steady vapor source rate systems should be much easier to measure than the typically less than 25% reduction in total number for an initial saturation ratio system.

Discussion

In order to explain the behavior of the system with an initial aerosol that may undergo a burst of nucleation, the interaction of the vapor source (if any) and the competing vapor removal processes of homogeneous nucleation and of condensational growth must be understood. This has been discussed for the constant source rate system (10,12), and will be expanded upon here.

The source-rate driven system begins with a vapor build-up phase, during which the vapor source totally dominates over the depletion mechanisms. If homogeneous nucleation is to occur, and the source rate is not exceptionally large relative to the intrinsic molecular collision rate of the saturated vapor, then the rise of the saturation ratio will be halted when the condensation rate onto supercritically sized particles becomes larger than the source rate. Since the aerosol is constantly growing larger (and we assume particle deposition is slow on this time scale and may be neglected), the saturation ratio rapidly falls with the rising potential for condensation onto aerosol per unit supersaturation. Homogeneous nucleation, with its very high dependence on the saturation ratio, thus occurs as a burst around the time of the peak saturation. The aerosol onto which condensation occurs may be either the initial aerosol or that generated by homogeneous nucleation. Either way (for dimensionless source rates of order unity or less), the dominant vapor depletion process is condensation, and any homogeneously nucleated aerosol rapidly grows quite large compared to the monomer size and to the nucleation time scale. The number of particles needed to quench nucleation depends on the dimensionless source rate; as the dimensionless source rate approaches zero, a single particle would be a sufficient condensation site to reverse the buildup of vapor before any additional nucleation occurred. In fact, the amount of "overshoot" in the saturation ratio after the first nucleation event occurs depends to a great extent on the dimensionless source rate. It also depends on the local sensitivity of the nucleation rate to the saturation ratio, since as the saturation ratio rises, the nucleation rate rises sharply, thus creating additional particles that can reverse the saturation buildup by acting as additional condensation nuclei.

From this dynamic viewpoint, the influence of initial aerosol and other parameters can be explained. Clearly, for a given peak saturation ratio, less initial aerosol will be needed to suppress nucleation in the source-driven system than in one like the PSM where nucleation is driven by the initial saturation ratio. The build-up period in the source-driven system allows the initial aerosol to grow larger and be more effective as condensation nuclei, so smaller numbers are more effective, and larger numbers will actually reverse the vapor build-up before critical supersaturation and the accompanying significant nucleation rate can ever occur. For an initial saturation system, regardless of the initial aerosol concentration, some homogeneous nucleation must occur. (This applies when a simple nucleation expression, neglecting cluster scavenging, is used. The falling N_J/N_{J_0} tails for high initial aerosol concentrations in Fig. 5–7 demonstrate that nucleation would never be entirely suppressed, given these assumptions. The -1 slope on the log–log scale of these tails is expected, since the condensation rate per unit supersaturation for particles of a fixed size is directly proportional to their number. For very high number, these particles can deplete the vapor without changing in size significantly, and the time duration of the nucleation burst, and hence the number of particles nucleated, is simply inversely proportional to the initial condensation rate.) The suppression of nucleation does not become complete for initial relative numbers larger than unity, as it would for the source-rate driven system, because the vapor build-up phase is missing.

For either type of system, for the same relative number of initial aerosol, a larger initial diameter will increase the condensation rate and thus lead to less homogeneous nucleation overall, explaining the main feature of Fig. 5. Suppression of nucleation is greater for higher initial saturation ratios in Fig. 6 because the higher saturation ratios lead to a longer period of significant nucleation rate. The nucleation rate is not only higher for larger S, but it shows a weaker sensitivity to S, i.e., $d\,ln\,J/d\,ln\,S$, which is a monotonically increasing function of the dimensionless activation energy, $0.5(\sigma^\star)^3/(ln\,S)^2$, which is smaller for larger values of S. The broader the range of S over which the nucleation rate stays significant, the more time there is for initial aerosol to grow and become better suppressors of the vapor saturation ratio and thus the overall amount of nucleation. In Fig. 7, the increased suppression of nucleation with higher temperature is due to the lowered dimensionless surface tension and thus lowered nucleation energy barrier, acting in the manner just described.

A qualitative consideration of the dynamics of the nucleation burst has explained the basic behavior noted in our simulations. Using some approximations, one may derive the approximate dependence of N_{J_0} on the system parameters (see Appendix B). The resulting relationships,

$$N_{J_0} = \begin{cases} 2.58\left[\frac{J_0}{\alpha(S_0-S_*)}\right]^{0.75}\left(\frac{S_0}{P}\right)^{0.25}, & \text{for } Kn >> 1; \\ 1.26\left[\frac{J_0}{\alpha(S_0-S_*)Kn^*}\right]^{0.60}\left(\frac{S_0}{P}\right)^{0.40}, & \text{for } Kn << 1. \end{cases} \qquad [13]$$

are confirmed by numerical simulations using condensation rate expressions appropriate for the limiting regimes. Generally, the controlling transport regime refers to condensable vapor transport to the particle during the nucleation burst itself. For the PSM experiments, the nucleation burst occurred with particles somewhere in the transition regime, so a power dependence of between 0.60 and 0.75 on initial saturation ratio would be expected. Table III shows simulation sensitivities (based on finite difference calculations) for N_{J_0} to the major system parameters, where the sensitivity X_p to parameter p is defined as $\mathrm{d}\,log\,N_{J_0}/\mathrm{d}\,log\,p$, evaluated for the selected test case. The sensitivity, or the local power dependence, of N_{J_0} to a ratio of nucleation to relative condensation rate is between 0.60 and 0.75, as expected from Eq. 13. The constant vapor source case shows a much weaker sensitivity to the nucleation and condensation expressions, as mentioned previously. Note that the initially supersaturated system's N_{J_0} shows a tremendous sensitivity to $\sigma^\star$ simply because J_0 shows an even larger sensitivity to $\sigma^\star$.

One interesting other feature of the simulations is the relative independence of the number suppression curve of Fig. 8 on the magnitude of the dimensionless source rate. The slower the source rate, the larger the initial aerosol can get, so the more effective it will be as condensation sites by the time homogeneous nucleation begins. Conversely, the higher source rate case leads to higher supersaturation ratios and much more nucleation in the absence of initial aerosol, so that the initial aerosol (having a much higher absolute number concentration but the same relative number concentration) has more time to grow (and at higher supersaturations) during the burst of nucleation than it would under lower source rates. The effects seem to partially offset each other, so lower dimensionless source rates lead to only slightly more effective suppression of nucleation by initial aerosol.

Conclusions

We have investigated the effect of initial aerosol on homogeneous nucleation for a system having a high initial supersaturation, and found such a system has important similarities and differences to one in which nucleation results from a constant source rate of condensable vapor. In both systems, if there is relatively little initial aerosol, the resulting number of particles is virtually the same as would have resulted in the absence of initial particles; when there is a relatively large number of initial particles, homogeneous nucleation is greatly suppressed, although much more thoroughly for the system driven by a source rate. For intermediate conditions, with the initial aerosol concentration around or just below the number concentration that would have resulted with no initial aerosol, the behavior is interesting and different for the two systems. In the initial saturation ratio system, a modest suppression of nucleation occurs, experimentally verified in the present work but near the detection limit of the PSM. In a constant source rate system, homogeneous nucleation is virtually entirely suppressed if sufficient initial aerosol is present, and the resulting aerosol is many times less in number concentration than that which would have resulted in the absence of initial aerosol. The difference in behavior lies in the buildup period of the source rate system, where the initial aerosol may grow large enough to deplete the condensable vapor faster than the source rate can increase it, and thus prevent the burst of homogeneous nucleation that occurs with sufficiently high supersaturations.

Acknowledgment

This work was supported by National Science Foundation Grant ATM-8503103 and Japan Grant-in-Aid for Environmental Science 60030039.

References

1. Okuyama, K., Kousaka, Y., Warren, D.R., Seinfeld, J.H., and Flagan, R.C., *Aerosol Sci. Technol.* **x**, xx (1986).
2. Higuchi, W.I. and O'Konski, C.T., *J. Colloid Sci.* **15**, 14 (1960).
3. Amelin, A.G., Vishnepolskaya, I.V., and Belyakov, M.I., *J. Aerosol Sci.* **2**, 93 (1971).
4. Rybin, E.N., Pankratova, M.E., and Kogan, Y.I., *Russ. J. Phys. Chem.* **51**, 617 (1977).
5. Anisimov, M.P. and Cherevko, A.G., *J. Aerosol Sci.* **16**, 97 (1985).
6. Kousaka, Y., Okuyama, K., Niida, T., Hosokawa, T, and Mimura, T., *Part. Charact.* **2**, 119 (1985).
7. Okuyama, K., Kousaka, Y., Motouchi, T., *Aerosol Sci. Technol.* **3**, 353 (1984).
8. Kousaka, Y., Niida, T., Okuyama, K., and Tanaka, H., *J. Aerosol Sci.* **12**, 231 (1982).
9. Knutson, E.O., and Whitby, K.T., *J. Aerosol Sci.* **6**, 443 (1975).
10. Warren, D.R., and Seinfeld, J.H., *J. Colloid Interface Sci.* **105**, 136 (1985).
11. Stern, J.E., Wu, J.J., Flagan, R.C., and Seinfeld, J.H., *J. Colloid Interface Sci.* **xx** xxx (1986).
12. Warren, D.R., and Seinfeld, J.H., *Aerosol Sci. Technol.* **3**, 135 (1984).

Appendix A. The Cell Model

Previous work by Stern et al. (11) has shown how to modify the SNM model to include steady state radial vapor profiles around each particle. The difference between this cell model approach and the simple SNM model will be reassessed here.

Assuming that each cell extends from the particle radius a to the cell boundary L in a space filling manner, the steady state vapor profile may be expressed as a function of the dimensionless radial distance r, where $\varepsilon \leq r \leq 1$ and $\varepsilon = a/L$. Letting S_ε be the equilibrium saturation ratio at the particle surface (by the Kelvin effect, $S_\varepsilon \geq 1$), and S_L be the cell boundary saturation ratio, and neglecting Stefan flow,

$$S_r = S_L \left[1 - \left(\frac{\varepsilon}{r}\right)\left(\frac{1-r}{1-\varepsilon}\right)\left(\frac{S_L - S_\varepsilon}{S_L}\right)(1 - f(Kn))\right] \qquad [A1]$$

This steady state radial vapor profile allows one to evaluate a spatial average nucleation rate $\bar{J}$ which is defined as

$$\bar{J} = \frac{\int_\varepsilon^1 J(S_r) r^2 \mathrm{d}r}{\int_0^1 r^2 \mathrm{d}r} \qquad [A2]$$

which may be different from the SNM model's assumed nucleation rate $J(\bar{S})$ based on the spatial average saturation ratio $\bar{S}$ given by

$$\bar{S} = \frac{\int_\varepsilon^1 S_r r^2 \mathrm{d}r}{\int_0^1 r^2 \mathrm{d}r} \qquad [A3]$$

Using these cell model equations, we now consider a worst case that maximizes spatial variations in S and hence J, that of continuum regime diffusion with negligible surface vapor pressure, gives

$$\bar{J} = 3\int_\varepsilon^1 J\left(S_L\left[1 - \frac{\varepsilon}{r}\frac{1-r}{1-\varepsilon}\right]\right) r^2 \mathrm{d}r \qquad [A4]$$

If we wish to consider cases where suppression of nucleation will be relatively small, we can approximate $J(S)$ by a power series expansion of $\log J$ in terms of $\log S$

around S_L. This gives

$$J(S_r) \approx J(S_L)\left(\frac{S_r}{S_L}\right)^P \qquad [A5]$$

where $P = 2 + [\sigma^\star/ln\, S_L]^3$. A further expansion of the expression of the form $(1-x)^P \approx 1 - xP$ for $xP \ll 1$ yields

$$\bar{J} \approx 3J(S_L)\int_\varepsilon^1 \left[r^2 - \frac{P\varepsilon}{1-\varepsilon}(r-r^2)\right] \mathrm{d}r \qquad \text{for} \quad \varepsilon P \ll 1 \qquad [A6]$$

Evaluating this integral and keeping only terms of at least order ε^2 gives

$$\frac{\bar{J}}{J(S_L)} \approx 1 - 0.5\frac{\varepsilon P}{1-\varepsilon} \qquad [A7]$$

$\bar{S}$ evaluates to a similar expression (using P=1), namely

$$\bar{S} \approx S_L\left[1 - 0.5\frac{\varepsilon}{1-\varepsilon}\right] \qquad [A8]$$

Interestingly, when $J(\bar{S})$ is now approximated for small ε, it has the same expression to order εP as does $\bar{J}$. Written as power series expansions of ε to two terms,

$$\frac{J(\bar{S})}{J(S_L)} \approx 1 - 0.5\varepsilon P - 0.5\varepsilon^2 P \approx \frac{\bar{J}}{J(S_L)} \qquad [A9]$$

Because $\bar{J}/J(S_L)$ represents a norm of order P for the saturation ratio in the cell, while $J(\bar{S})/J(S_L)$ represents a norm of order one for the saturation ratio in the cell, and $P > 1$, the nucleation rates must be ordered $J(\bar{S}) \leq \bar{J} \leq J(S_L)$. Hence, for small εP, the ratio of nucleation rates used in the SNM and steady state cell models is bounded by

$$1 - 0.5\varepsilon P < \frac{J(\bar{S})}{\bar{J}} \leq 1 \qquad [A10]$$

Although the lower bound is likely to be a considerable underestimate of the ratio between the nucleation rates of the two models, this still establishes that the SNM and steady state cell models will agree whenever $\varepsilon P \ll 2$.

To obtain a more powerful condition for model agreement, numerical integration is required to evaluate Eq. A4 for cases where εP may approach or exceed unity. Table IV shows the ratio of the nucleation expressions over a wide range of εP (assuming Eq. A5 applies). Eq. A9 is confirmed for values of εP less than unity. When εP equals one, the difference between the two average nucleation rates is under 20%, and $\varepsilon P < 0.2$ leads to less than a 1% difference between the average nucleation rates and less than a 10% suppression of nucleation beneath the rate at the cell boundary. Hence spatial variations have an insignificant impact on the nucleation rate whenever $\varepsilon P < 0.5$. Since P, which is equal to the critical cluster number plus two, is typically of order 20 and under 100 (as $W_c^\star = 2P \ln S_L$, and $W_c^\star > 50$ makes nucleation incredibly slow) for just about any conceivable case with a nucleation rate fast enough to be of interest, the above requirement will be satisfied whenever $L/a > 100$ or the volume fraction of aerosol is less than 10^{-6} (a 1 g m^{-3} mass loading of unit density particles). In the case of the PSM experiments, P is close to 18 and ϵ varies from 0.0002 to 0.002, indicating that the cell model predictions will be indistinguishable from those of the SNM model for any of these simulations.

Hence for dilute systems of condensable vapor, spatial variations of the vapor concentration have a negligible effect on the average nucleation rate. For high volume fractions of aerosol, if nucleation is significant it must be relatively fast, hence leading to a relatively low value of P (and if $P = 20$, spatial inhomogeneities will have a 20% effect on nucleation rates only for mass loadings in excess of 125 g m^{-3}) and quite possibly to unsteady state nucleation. For the steady state cell model to be useful, one would have to be dealing with a high vapor pressure substance at high mass loadings, which still obeys the steady state nucleation expression.

Appendix B. Approximate Behavior of the SNM Model

In order to gain greater insight into the dynamics of nucleation and condensation for a system with a high initial saturation ratio, one may look for a simplified description of system behavior based on the results of the SNM model.

Nucleation occurs as a burst, with an almost constant rate for early times, which then drops off fairly rapidly. So we may approximate nucleation as an on/off phenomenon which ends at some time τ_N. Thus, for the case of no seed aerosol,

$$N = \begin{cases} J_0\tau, & \text{if } 0 \le \tau \le \tau_N; \\ J_0\tau_N, & \text{if } \tau > \tau_N. \end{cases} \qquad [B1]$$

Since the saturation ratio is assumed constant during this period of duration τ_N, it is possible to use Eq. 5 for the rate of change in mass with time (the mode subscript is dropped since only mode 2 is present),

$$\mathrm{d}M/\mathrm{d}\tau = \begin{cases} \alpha\,(S - S_\epsilon)\,N^{1/3}M^{2/3}, & \text{for } Kn >> 1; \\ \alpha\,(S - S_\epsilon)\,N^{2/3}M^{1/3}\left(\frac{4}{3}Kn^\star\right), & \text{for } Kn << 1. \end{cases} \qquad [B2]$$

Substituting in Eq. B1 for the aerosol number gives the following solutions for the two transport regimes:

$$M = \begin{cases} \left(0.25(S_0 - S_\epsilon)\right)^3 J_0\tau^4, & \text{for } Kn >> 1; \\ \left(\frac{8}{15}\alpha(S_0 - S_\epsilon)Kn^\star\right)^{3/2} J_0\tau^{5/2}, & \text{for } Kn << 1. \end{cases} \qquad [B3]$$

Since we have expressions for N and M for $\tau \le \tau_N$, we now need to determine τ_N such that

$$J_0\tau_N = \int_0^\infty J(S_\tau)\mathrm{d}\tau \qquad [B4]$$

To keep matters simple, we can satisfy Eq. B4 to within a few percent by τ_N to be the dimensionless time at which $J/J_0 = 0.5$, since the nucleation cutoff is relatively abrupt. We shall evaluate J/J_0 using the previously introduced power series linearization for nucleation as a function of S,

$$\frac{J}{J_0} \approx \left(\frac{S}{S_0}\right)^P \qquad [B6]$$

where $P = 2 + (\sigma^\star/ln\,S_0)^3$. We will find τ_N such that $J/J_0 = 0.5$ by Eq. B6 after integrating Eq. 1 for S_τ using the approximate expressions for N and M found earlier. Assuming $C \gg J$,

$$\frac{dS}{\mathrm{d}\tau} = \begin{cases} -\alpha\,(S - S_\epsilon)\,N^{1/3}M^{2/3}, & \text{for } Kn >> 1; \\ -\alpha\,(S - S_\epsilon)\,N^{2/3}M^{1/3}\left(\frac{4}{3}Kn^\star\right), & \text{for } Kn << 1. \end{cases} \qquad [B7]$$

Since τ_N implies that $(S/S_0) = \sqrt[P]{0.5}$, we will find τ_N when

$$\Delta S/S_0 = 1 - \sqrt[P]{0.5} = 1 - e^{-(ln\,2)/P} \approx \frac{0.69}{P} \qquad [B8]$$

Solving for S in Eq. B7, under the assumption that $(S - S_\epsilon)$ equals $(S_0 - S_\epsilon)$ (the same results for N_{J_0} may be obtained by integrating under the assumption that $(S - S_\epsilon) = S$ and later linearizing for small $\Delta S/S_0$),

$$S_0 - S = \begin{cases} \alpha^3 J_0 {S_0}^3 {\tau_N}^4/64, & \text{for } Kn >> 1; \\ 0.39\left[\alpha S_0 Kn^\star\right]^{1.5} J_0 {\tau_N}^{2.5}, & \text{for } Kn << 1. \end{cases} \qquad [B9]$$

Solving for the dimensionless nucleation duration, τ_N, in the above expression, for $S_0 - S = 0.69 S_0/P$, and then using $N_{J_0} = J_0 \tau_N$ gives us expressions for the resulting dimensionless aerosol number formed in the absence of seed particles,

$$N_{J_0} = \begin{cases} 2.58\left[\frac{J_0}{\alpha(S_0 - S_\epsilon)}\right]^{0.75}\left(\frac{S_0}{P}\right)^{0.25}, & \text{for } Kn >> 1; \\ 1.26\left[\frac{J_0}{\alpha(S_0 - S_\epsilon)Kn^\star}\right]^{0.60}\left(\frac{S_0}{P}\right)^{0.40}, & \text{for } Kn << 1. \end{cases} \qquad [B10]$$

The most noteworthy factor in the above expressions for the number of particles nucleated is the bracketed ratio of initial nucleation rate to a portion of the condensation rate expression, raised to a power somewhat less than unity. These power dependences were exactly confirmed by SNM simulations using condensation expressions for the kinetic and continuum limits, respectively. Evaluating the expressions in Eq. B10 for a typical experiment gave values which were very close (1% low and 8% low for large and small Knudsen numbers, respectively) to the SNM predicted N_{J_0} using the appropriate limiting regime condensation expressions. Furthermore, the SNM model showed aerosol number concentrations rising linearly with time up until almost τ_N, and mass concentrations which showed the power dependence with time as predicted by Eq. B3 until τ approached τ_N. Thus the behavior of the SNM model for initial saturation driven systems is well understood, for limiting regime cases.

In nucleating systems at atmospheric pressure, the mean aerosol diameter usually passes through the transition regime during the burst of nucleation, so that the kinetic growth expression may apply to early times and the continuum growth expression to later times. Since each limiting regime expression will overpredict the growth rate of particles which actually lie outside the size regime of applicability, assumption of either limiting regime expression will overpredict the depletion of vapor during the nucleation burst, thus underpredicting τ_N and N_{J_0}. For our typical DBP experiment, assumption of either limiting regime growth expression led to about 30% fewer particles nucleated than use of the general growth expression, during an otherwise identical simulation using the SNM model.

Table I: Measured PSM Number Concentrations

RUN	Temp	S	N_i	N_{J_0}	N_f	N_i/N_{J_0}	N_J/N_{J_0}	Suppression	
	°C		cm^{-3}	cm^{-3}	cm^{-3}			N_J	N_f
A2	42.29	178.13	425,400	95,240	533,300	4.467	1.133	−13%	−2%
A3	42.78	190.05	425,400	292,100	622,200	1.456	0.674	33%	13%
A4	43.14	194.15	425,400	393,700	749,200	1.081	0.822	18%	9%
A5	43.43	199.66	425,400	527,000	812,700	0.807	0.735	27%	15%
A6	43.71	205.24	425,400	1,251,000	1,492,000	0.340	0.853	15%	11%
A7	43.92	212.91	425,400	2,216,000	2,476,000	0.192	0.925	7%	6%
B2	43.01	163.39	596,800	31,750	685,700	18.797	2.800	−180%	−9%
B3	43.42	176.53	596,800	133,300	742,900	4.477	1.096	−10%	−2%
B4	43.62	183.37	596,800	444,400	844,400	1.343	0.557	44%	19%
B5	43.75	192.21	596,800	831,700	1,270,000	0.718	0.809	19%	11%
B6	43.95	199.49	596,800	666,700	946,000	0.895	0.524	48%	25%
B7	44.08	208.92	596,800	2,005,000	2,279,000	0.298	0.839	16%	12%
B8	43.87	201.39	596,800	863,500	1,283,000	0.691	0.795	21%	12%
B9	43.67	194.04	596,800	368,300	787,000	1.620	0.516	48%	18%
C2	33.36	274.14	349,200	76,190	419,000	4.583	0.916	8%	2%
C3	33.65	299.16	349,200	215,900	495,200	1.617	0.676	32%	12%
C4	33.85	310.47	349,200	558,700	730,200	0.625	0.682	32%	20%
C5	34.04	322.08	349,200	761,900	730,200	0.458	0.500	50%	34%
C6	34.23	334.00	349,200	1,848,000	1,943,000	0.189	0.862	14%	12%
C7	34.33	350.29	349,200	2,946,000	2,965,000	0.119	0.888	11%	10%
D2	31.36	314.87	561,900	95,240	673,000	5.900	1.167	−17%	−2%
D3	31.65	343.96	561,900	279,400	730,200	2.011	0.602	40%	13%
D4	32.02	348.64	561,900	368,300	806,300	1.526	0.664	34%	13%
D5	32.30	357.53	561,900	552,400	958,700	1.017	0.718	28%	14%
D6	32.41	375.38	561,900	857,100	1,251,000	0.656	0.804	20%	12%
D7	32.51	393.95	561,900	1,143,000	1,410,000	0.492	0.742	26%	17%
D8	32.61	413.25	561,900	2,673,000	2,692,000	0.210	0.797	20%	17%

Table II: Predicted PSM Number Concentrations Using $E_J = 10^7$

RUN	N_{J_0}	N_f	N_{J_0}/N_{J_0}	N_i/N_{J_0}	N_J/N_{J_0}	Suppression	
	cm^{-3}	cm^{-3}	Pred:Meas			N_J	N_f
A2	78,560	443,600	0.82	5.415	0.232	77%	12%
A3	243,600	527,900	0.83	1.746	0.421	58%	21%
A4	396,700	633,200	1.01	1.072	0.524	48%	23%
A5	671,000	852,700	1.27	0.634	0.637	36%	22%
A6	1,118,000	1,247,000	0.89	0.381	0.735	27%	19%
A7	1,990,000	2,064,000	0.90	0.214	0.823	18%	15%
B2	39,140	602,300	1.23	15.248	0.141	86%	5%
B3	139,300	636,800	1.05	4.284	0.287	71%	13%
B4	256,500	697,900	0.58	2.327	0.394	61%	18%
B5	504,600	867,000	0.61	1.183	0.535	46%	21%
B6	906,500	1,197,000	1.36	0.658	0.662	34%	20%
B7	1,736,000	1,953,000	0.87	0.344	0.781	22%	16%
B8	972,400	1,254,000	1.13	0.614	0.676	32%	20%
B9	541,800	895,400	1.47	1.102	0.551	45%	21%
C2	69,390	365,800	0.91	5.032	0.239	76%	13%
C3	233,600	453,500	1.08	1.495	0.446	55%	22%
C4	406,500	578,400	0.73	0.859	0.564	44%	23%
C5	696,500	818,800	0.91	0.501	0.674	33%	22%
C6	1,182,000	1,257,000	0.64	0.295	0.768	23%	18%
C7	2,148,000	2,168,000	0.73	0.163	0.847	15%	13%
D2	102,900	588,300	1.08	5.461	0.257	74%	12%
D3	340,700	719,800	1.22	1.649	0.463	54%	20%
D4	498,700	833,100	1.35	1.127	0.544	46%	21%
D5	786,100	1,065,000	1.42	0.715	0.640	36%	21%
D6	1,449,000	1,656,000	1.69	0.388	0.755	24%	18%
D7	2,622,000	2,764,000	2.29	0.214	0.840	16%	13%
D8	4,680,000	4,772,000	1.75	0.120	0.900	10%	9%

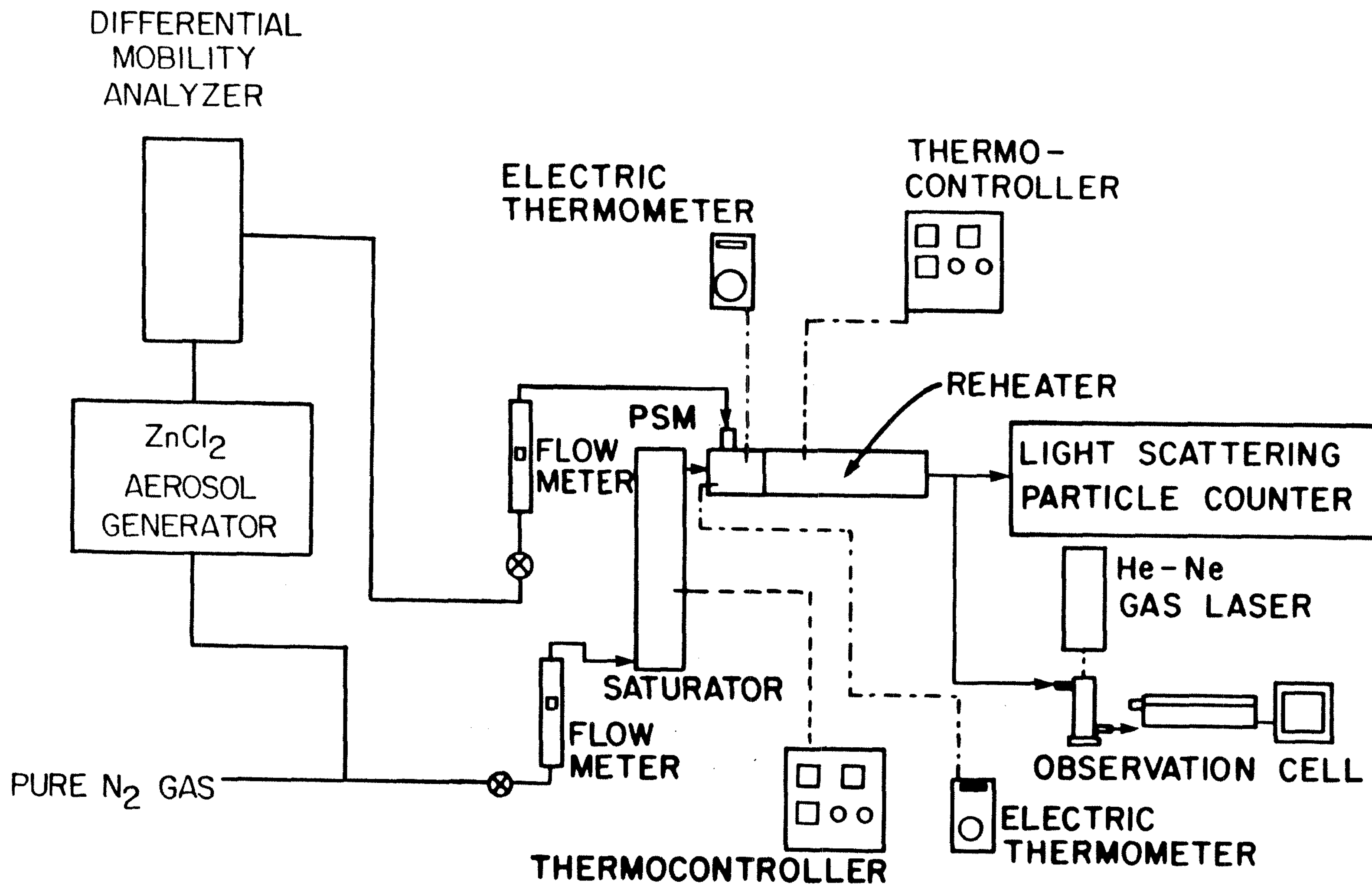
Table III: Sensitivity of N_{J_0} to Input Parameters for the SNM Model

	Value	Sensitivity X_p	
	of Input	Case I	Case II
	Parameter†	S_0 driven	$R^\star$ driven
S_0	250	+11.9	*n.a.*
$R^\star$	0.1	*n.a.*	+ 1.33
$\sigma^\star$	13.921	−89.8	−13.2
$Kn^\star$	197.	− 0.31	− 1.30
α	1.0	− 0.67	− 1.42
E_J	10^7	+ 0.67	+ 0.092
g_s	500	$< 10^{-3}$	$< 10^{-4}$

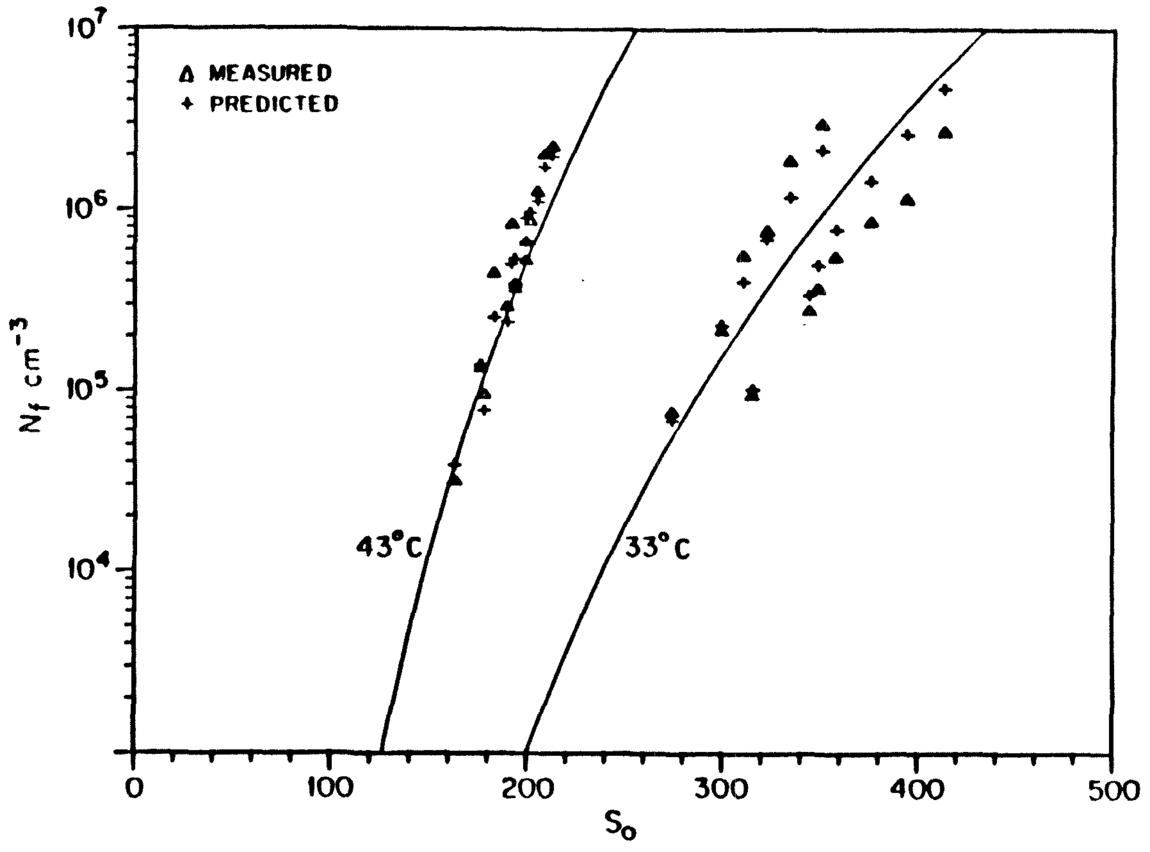
†Simulation is for DPB vapor at 40 °C with no initial aerosol.

Table IV: Spatial Average Nucleation Rates for Various ε and P

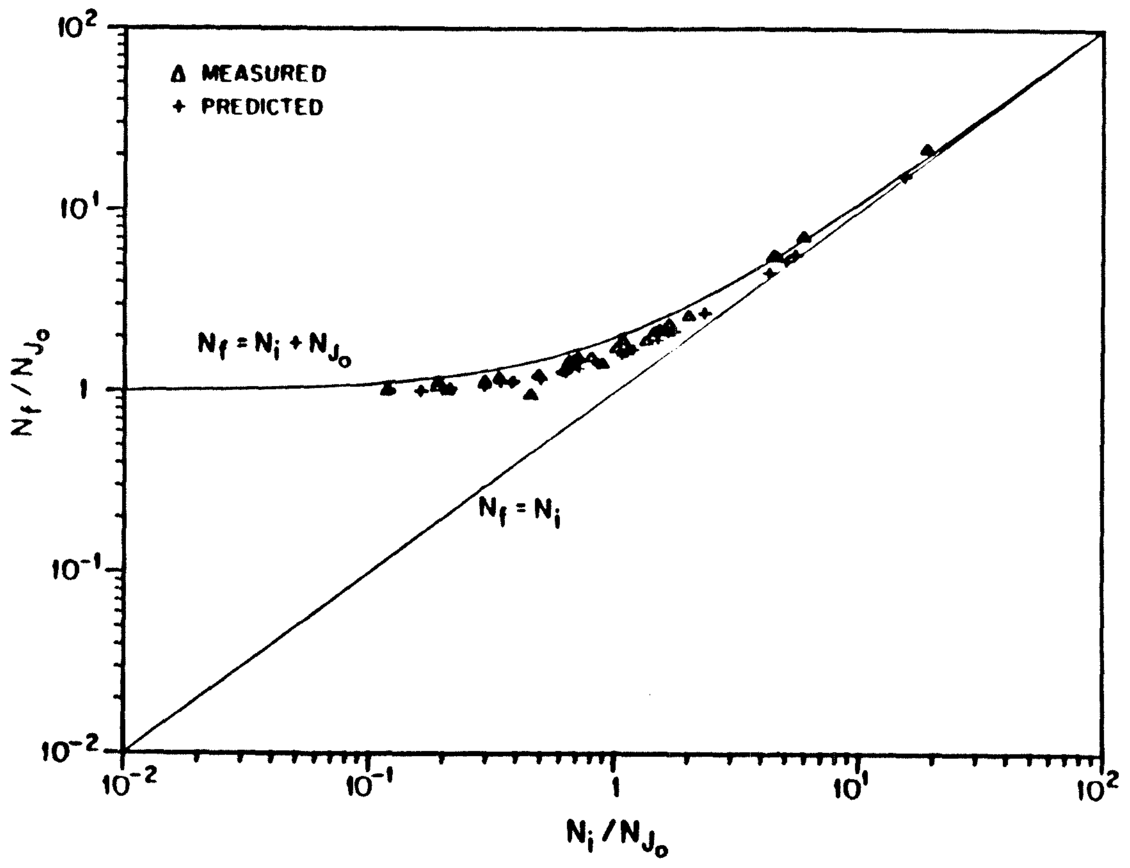
ε	P	εP	$\frac{J(\bar{S})}{J(S_L)}$	$\frac{\bar{J}}{J(S_L)}$	ΔJ %
0.0005	200	0.1	0.952	0.954	0.3 %
0.0025	200	0.5	0.779	0.817	4.9 %
0.005	200	1.0	0.605	0.701	15.8 %
0.0002	50	0.01	0.995	0.995	0.0 %
0.002	50	0.1	0.951	0.953	0.3 %
0.004	50	0.2	0.905	0.914	1.0 %
0.006	50	0.3	0.860	0.878	2.1 %
0.01	50	0.5	0.776	0.815	5.0 %
0.02	50	1.0	0.599	0.697	16.3 %
0.04	50	2.0	0.350	0.543	55.4 %
0.10	50	5.0	0.059	0.324	448 %
0.01	10	0.1	0.951	0.953	0.3 %
0.05	10	0.5	0.766	0.807	5.3 %
0.10	10	1.0	0.568	0.674	18.7 %



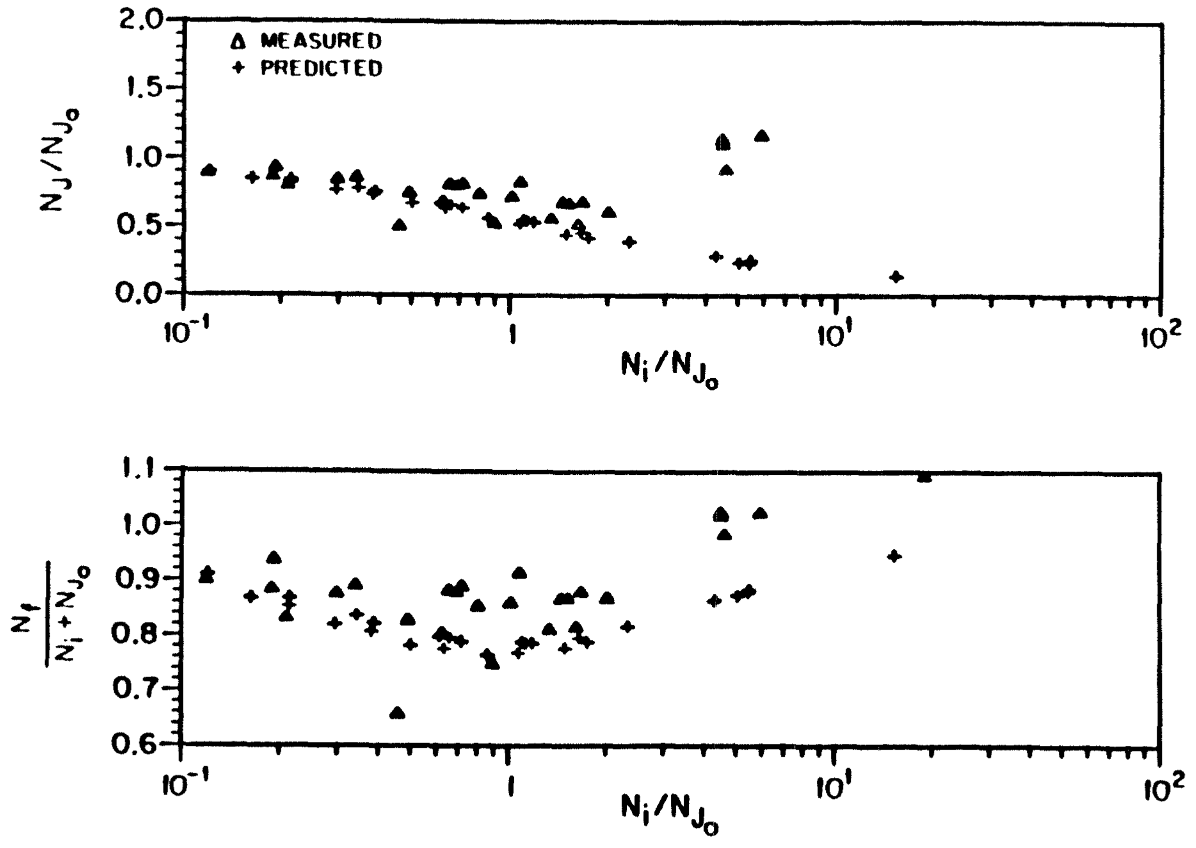
1. Schematic diagram of the experimental apparatus.



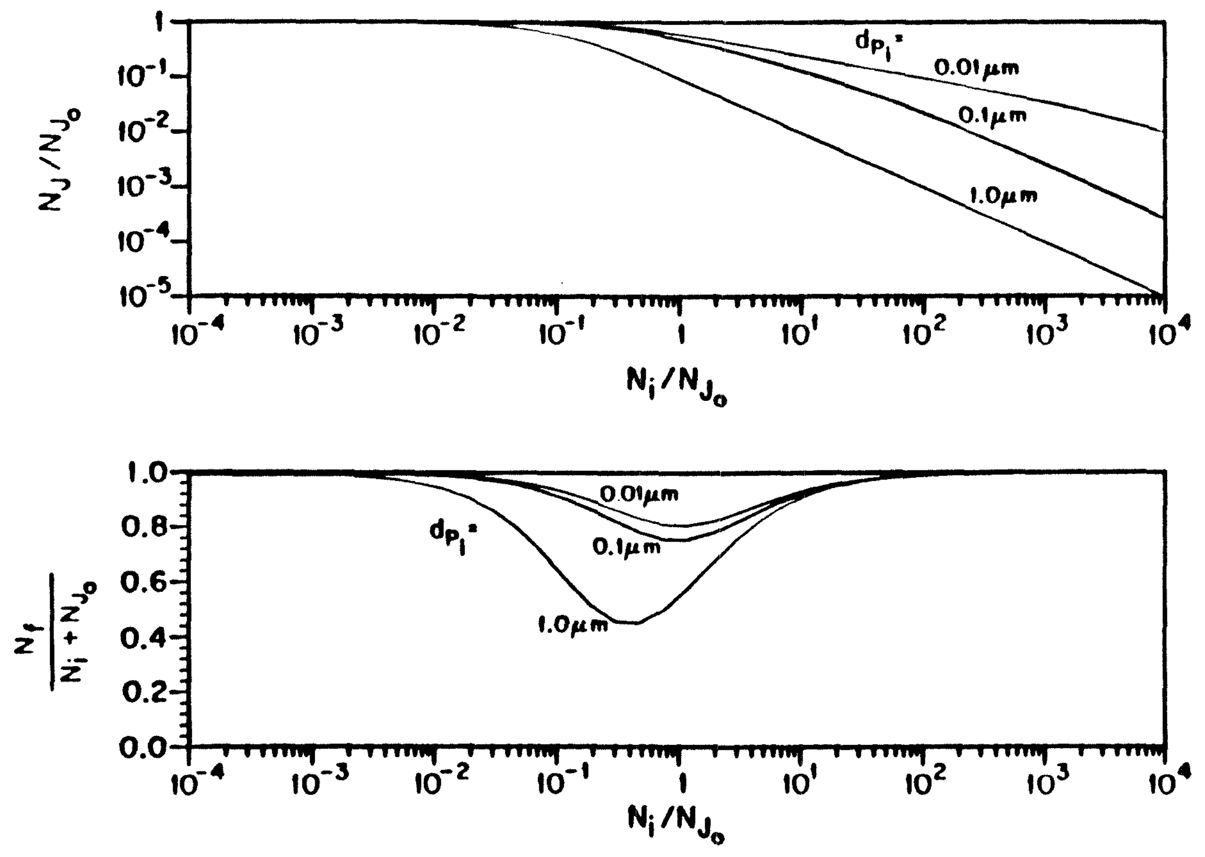
2. Measured and simulated aerosol number concentrations generated in the PSM by homogeneous nucleation of dibutylphthalate (DBP) in the absence of initial aerosol.



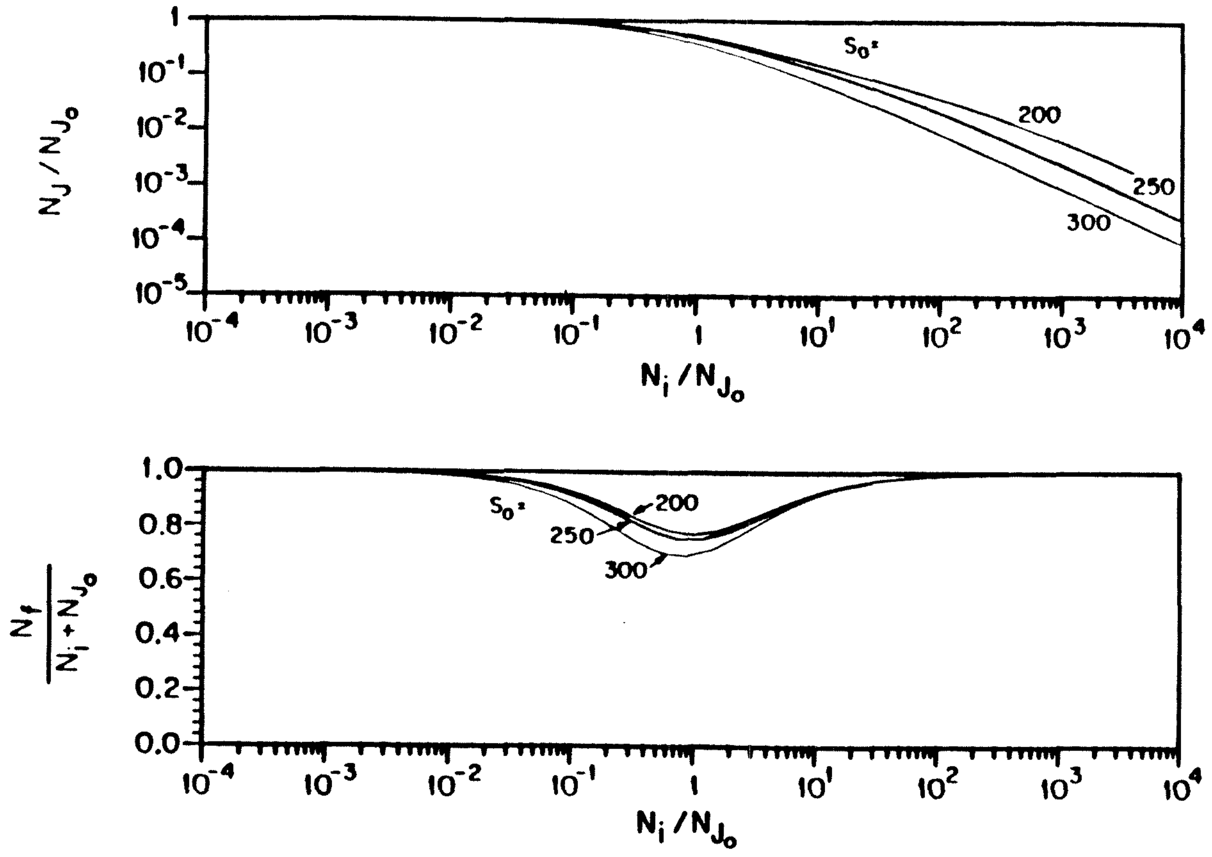
3. Measured and simulated relative aerosol number concentrations generated in the PSM as a function of relative initial aerosol number.



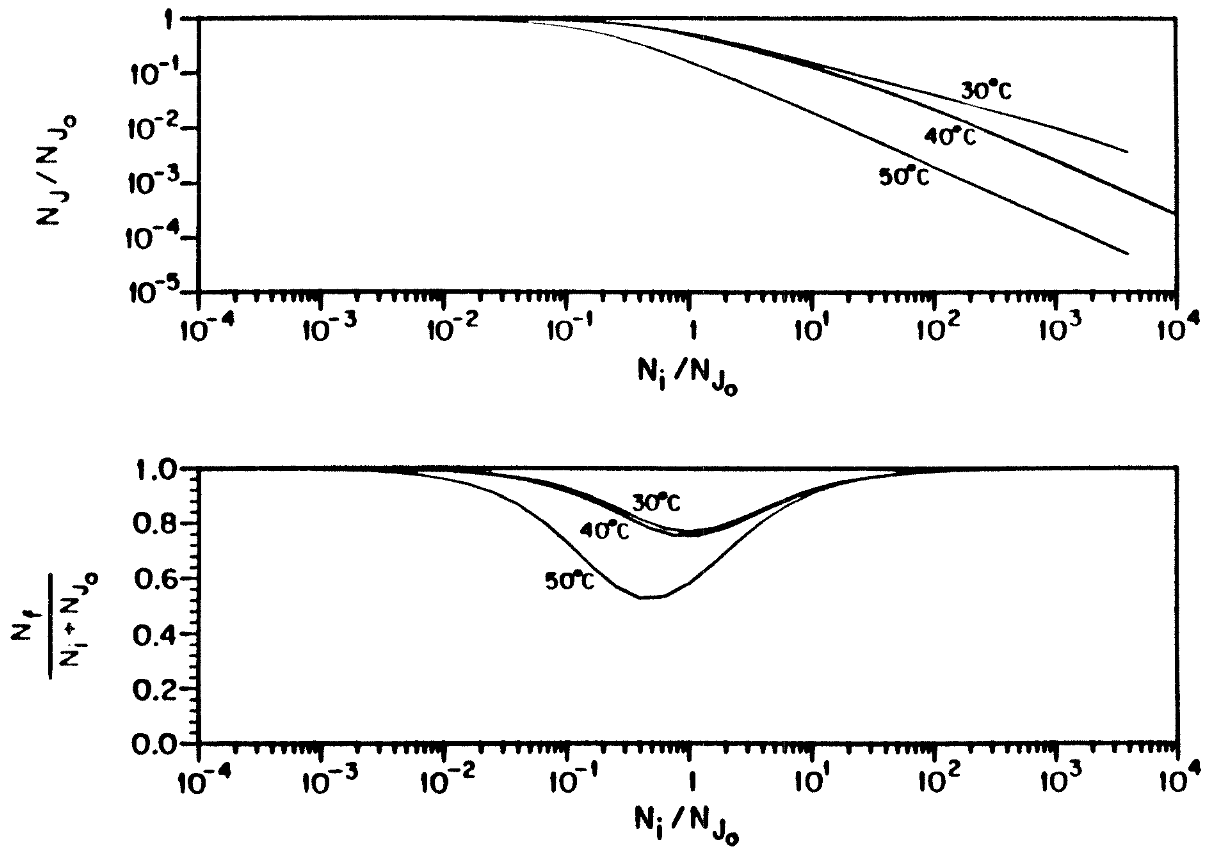
4. Measured and simulated suppression of overall nucleation and number concentration due to initial aerosol.



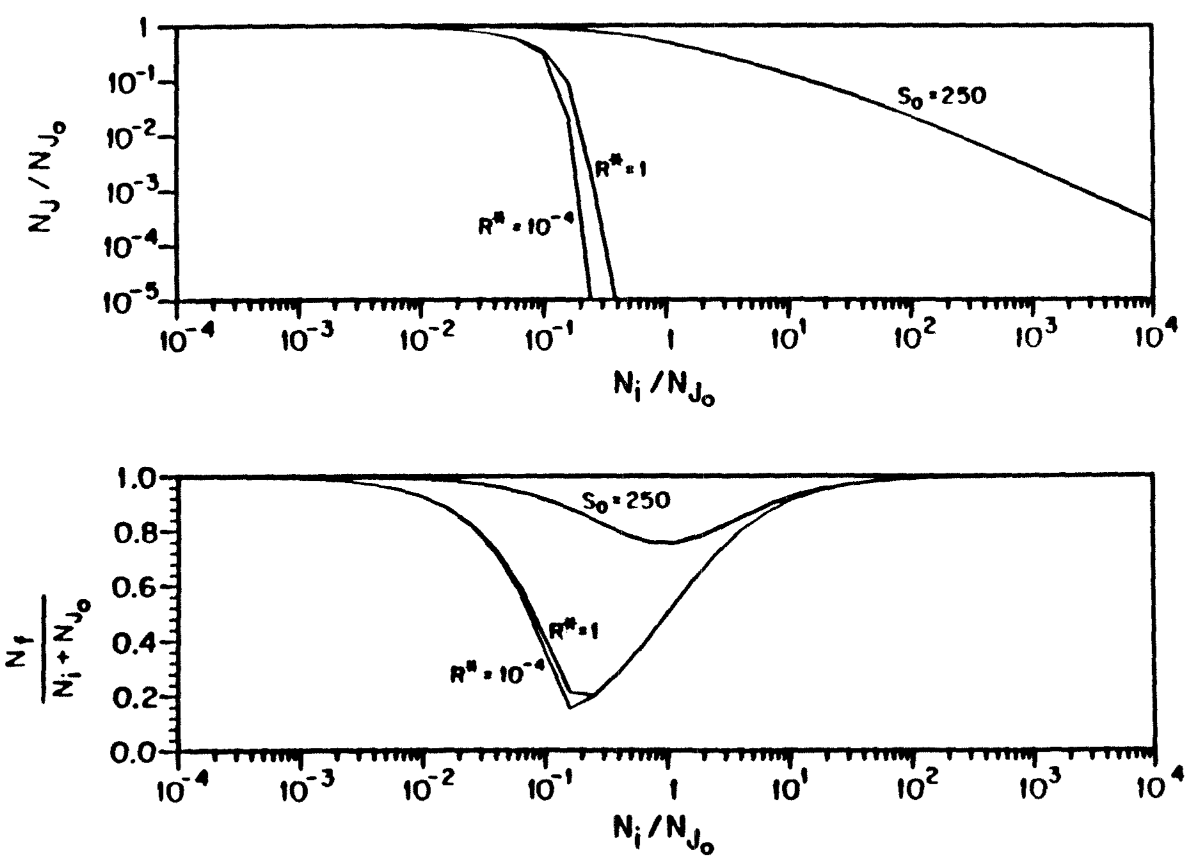
5. Simulated suppression of overall DBP nucleation and total number due to initial aerosol for different initial particle sizes at $S_0 = 250$ and 40 °C.



6. Simulated suppression of overall DBP nucleation and total number due to initial aerosol for different initial saturation ratios for 0.1 μm particles at 40 °C.



7. Simulated suppression of overall DBP nucleation and total number due to initial aerosol for different temperatures for 0.1 μm particles and $S_0 = 250$.



8. Simulated suppression of overall DBP nucleation and total number due to initial aerosol for two vapor-source-rate-driven systems and the PSM initial-saturation-ratio-driven system.

CHAPTER 7:

TOLUENE PHOTOCHEMICAL AEROSOL EXPERIMENTS IN AN OUTDOOR TEFLON SMOG CHAMBER

Introduction

Outdoor teflon smog chambers have been used extensively to gain information about the photochemical reactions which occur in the atmosphere, especially in polluted urban atmospheres. The world-renowned LA smog is an outstanding example of the phenomenon of interest. The Los Angeles basin endures with prominent photochemical smog because the area is blessed by several natural and anthropogenic factors which encourage atmospheric photochemistry: bright sunshine, high temperatures, a basin topology which traps air parcels inland between the sea and mountains, and a large and mobile population with accompanying industrial infrastructure. Smog chamber experiments allow us to observe and enhance and control the photochemical experiments which man and nature are jointly conducting around us.

In this chapter I will discuss the toluene photochemical aerosol experiments which were performed in the summer of 1985. These were a follow-up to toluene photochemistry experiments which were performed predominantly in the summer and fall of 1983 and which have been been discussed in great detail in Joe Leone's Ph.D. thesis, as well as by Leone et al. (1985). The 1983 experiments helped establish a chemical mechanism for the photochemical oxidation of toluene in the presence of oxides of nitrogen. Having a photochemical model for the system, it was viable to study how the photochemically generated condensable vapors formed aerosol particles. In particular, in view of previous theoretical and computer simulations of the competition between homogeneous nucleation and condensation, it would be interesting to see if seed aerosols could be shown to inhibit homogeneous nucleation in a real system–especially when the answer has such relevance to health and visibility as does photochemically generated organic aerosol.

Smog Chamber Construction

Our smog chambers are constructed of 0.002-inch thick teflon copolymer film. Thin teflon film is the standard smog chamber material because of teflon's inertness, transparency, toughness, and impermeability. A clean 2 mil film transmits 95% or more of visible and ultraviolet radiation, thus allowing us to expose the contents of the chamber to nearly the true solar spectrum. The smog chamber was constructed by heat sealing together two panels of teflon, which form a pillow configuration when inflated. For the full-size chamber in which our main runs were conducted, each of the teflon panels was constructed of five thirty-foot long strips of teflon off a four foot wide roll. The process of cutting and heat-sealing the teflon takes two or three people about a day to accomplish, depending on how much maneuvering space is available. A few more hours are required for reinforcing all seams with green mylar circuit-board tape, and installing the necessary ports for tubing into the bottom of the chamber. The resulting full-sized chamber is nearly 30 feet by 20 feet flat, and will hold about 65 cubic meters of air when fully inflated. This chamber is termed "full-sized" because it just about covers a steel and clothesline support structure which keeps the chamber two feet above (for air circulation and accessibility) the ground, which is actually the roof of Keck Laboratory on the Caltech campus. The Air Quality ("Roof") Lab from which the smog chamber experiments are run represents part of a partial fourth story.

Smog Chamber Characteristics

Because teflon smog chambers have been in use for many years, their influence on gas phase chemistry is reasonably well characterized, as discussed by Leone et al. (1985). They are not quite a perfect invisible box in which chemical reaction may proceed. Reactive organic compounds can deposit and then revolatilize off the walls, so that a day devoted to "baking out" between runs is a necessary precaution. Also, the teflon walls provide a source of free radicals which enhances the chemical reactivity of the system beyond what a free air parcel of the same initial composition would show. The wall radical source rate differs for different chambers and even with age for a given chamber.

The influence of teflon smog chambers on aerosol dynamics has not drawn as much attention as for gas phase dynamics. The loss mechanisms for particles in a mixed vessel are diffusional and gravitational, and their combined effect produces first order loss rate expression for number concentration that was theoretically calculated and experimentally verified in a large glass vessel by Crump and Seinfeld (1981). For sufficiently small particles, the turbulent diffusive loss term dominates, while the settling term dominates for sufficiently large particles, leaving a distinct minimum for particles of intermediate size (around 0.1 μm) where neither mechanism is fast. Unfortunately, as I found by attempting to apply the above mentioned expression to teflon smog chamber aerosol loss data taken by Daniel Grosjean around 1982, particles of approximately 0.1 to 0.3 μm diameter, which should show very low loss rates, disappear several times faster than predicted by diffusion and settling, even after correcting for coagulation and condensational growth.

Recently, McMurry and Rader (1986) have developed a theory which can explain the elevated wall loss rates, beginning with the observation that teflon tends to develop an electrostatic charge (which will be affirmed by anyone who has ever handled teflon smog chambers and received a static electric shock or noticed the amount of dust which they accumulate). Aerosol particles can develop a charge due to air ions, and charged particles will be removed quite rapidly by the typical electric field strength on teflon film. Unfortunately the typical electric field strength on the surface of a teflon bag is not readily calculable; the surface mean electric field can be shown to be zero, with local regions on the teflon assuming positive or negative charges. Humidity, friction due to motion caused by the wind, and temperature effects could have significant effects on teflon's proclivity to develop local electrostatic charges, leaving no a priori way to calculate particle loss rates accurately for the outdoor chamber, as evidenced by strong day to day variations in a preliminary seed aerosol loss rate study we conducted in our chamber. This makes modeling the aerosol data a bit more difficult; fortunately, the duration of homogeneous nucleation will be found to be on the order of thirty minutes or less, during which time deposition cannot have a great effect on the size distribution.

Gas Phase Instrumentation

In order to relate the production of aerosol to the gas phase chemistry, several gas phase measurements were routinely taken throughout each smog chamber run. The details of the gas phase instrumentation may be found in the thesis by Leone (1984). In summary, the gas phase instruments were the following:

1. a Dasibi Model 1008 PC for measuring ozone;
2. a Thermo Electron Model 14D/E chemiluminescent monitor for NO, NO_2, and NO_x;
3. a Hygrometix Model 8501 piezo-electric relative humidity meter;
4. thermistors in the inside manifold and the outside sample line;
5. an HP 5830 Gas Chromatograph with PID detectors for hydrocarbons, primarily toluene; and
6. Varian Model 1440 Gas Chromatograph with ECD Detector (Panalyzer) for peroxyacetyl nitrate.

Instruments 1–4 were continuously sampled and could be averaged by the data acquisition computer (although they were not during the aerosol runs for reasons to be explained later). The HP GC needed at least a four minute cycle for its toluene measurements, while the Panalyzer was on a 15 minute timer. All gas phase sampling equipment was located inside the roof lab, sampling from the central manifold.

Aerosol Instrumentation

Three types of instruments were available for nearly continuous measurements of the aerosol phase. These were the Electrical Aerosol Analyzer (EAA) by TSI, the water-based Condensation Nuclei Counter (CNC) by Enviroment One, and a Royco Laser Optical Particle Counter (OPC). In addition, end of run quartz filter samples were sometimes taken.

Two EAAs (Serial Number 132 and Serial Number 250) were available to us, as well as two CNCs. The EAAs could return size distribution data for particles from 0.0056 to 1.00 μm, although response at the extremes of the range was rather poor. The CNCs should be able to provide a total particle count of particles of roughly 0.005 μm and larger, with no useful discrimination by size. The laser OPC counts individual particles larger than 0.12

μm diameter. The 1983 toluene study had demonstrated that the aerosol would overload the few thousand particles cm^{-3} maximum of the OPC, and hence an approximately 100:1 dilution system was prepared for use with the OPC in 1985.

In the summer of 1984, a serious effort was made to check the calibration of the aerosol instruments. The electrometer of an EAA was used as a Faraday cage to measure the absolute flow of charged particles. An atomizer receiving a continuous feed from a syringe pump was used to generate small particles which, after charge neutralization, should contain very few multiply charged particles. These particles were fed to a differential mobility analyzer (DMA), so that an essentially singly charged and quite monodisperse aerosol was produced. The EAA electrometer thus was able to serve as an absolute standard for particle number, assuming the flow was measured accurately and the aerosol leaving the DMA was essentially all singly-charged. This aerosol stream was also fed into the other instruments, which allowed the calibration of the CNC's. It was found that the OPC agreed quite well with the EAA electrometer readings (using 0.2 μm diameter PSL). Occasional checks were made ever since to see that the pair of CNCs stayed in agreement with each other, and that the pair of EAAs did likewise.

The Data Acquisition Systems

The 1983 toluene smog chamber chemistry data was taken using a data acquisition system on the PDP11-03 running under the RT-11 ForeGround/BackGround operating system. This multichannel data acquisition system known as ASAP was originally written by Steve Heisler in 1975, documented by Pete McMurry in 1977, and modified by me in 1981 (as described in my candidacy report, "An Improved Computerized Data Acquisition System for Air Sampling Studies"). Unfortunately ASAP was half assembly language and thus difficult to modify, and ASAP did not perform continuous signal averaging but rather took a single reading for a channel each minute or so, thus being quite vulnerable to signal noise.

The decision was made to write an entirely new data acquisition system when it was realized that the quality of EAA data (with noise of around 5 mv) obtained from ASAP was not adequate to get consistent total number readings even when a human operator could average the signals by eye and get fairly steady aerosol number concentrations from the EAA. Certainly a proper continuous sampling program could do even better. ASAP was not amenable to such modifications, and generally seem outdated. Soon a new family of modular RT-11 sampling programs was born.

The new programs grew up around a multileveled package of Fortran subroutines (combined with a few necessary assembly language routines) that eventually came to be named RTLIB. This library of subroutines would handle VT100 terminal graphics and plotting, data acquisition, timing and time conversion, simple statistics, D/A output, and other conveniences for the RT-11 programmer. At last count, RTLIB had over 120 user callable subroutines. Full documentation may be found in Appendix C, along with a few of the data acquisition programs which utilized RTLIB.

The program which was to replace ASAP is called WATCH. Like ASAP, it provides up to 16 simultaneous A/D sampling channels for the users, with a compact data storage structure and allows the user to interactively tell the computer to start or stop sampling or recording a channel, to change ranges, or to enter a comment into the log. WATCH also provides a convenient, constantly updated video display and superior video plotting ability. Most importantly WATCH provides essentially continuous sampling of the analog channels, and recording of the time-averaged value. Unfortunately, WATCH also taxes the memory limits of our 56KB accessible RAM system, and under certain conditions (sometimes during range changes or during autoplotting) may cause the system to crash. (The overlay handler doesn't seem to work well with WATCH, although WATCH violates no overlay rules.) Because of these reasons, the EAA control routines were never added to WATCH. WATCH seems to work for multichannel analog instrument sampling, but since all recent roof lab experiments have involved aerosols (and the EAA), WATCH has become a program without an immediate use.

Several other RTLIB-based programs did come into regular use for the toluene aerosol experiments. The key program was DO2EAA, a program devoted to cycling a pair of EAA's while keeping the user fully informed of their progress, and saving a record on floppy diskette. (DOEAA is the version for a single EAA.) SAVEAA compresses the main EAA information into a smaller file, and VTEAA provides a VT100 display of the EAA channel profiles. SAVOPC is the most convenient way for a user to transfer the data from the OPC printout into a computer file, with a minimum of keystrokes. SAVTOL accomplishes the same feat for entering gas chromatograph peaks from chart paper. These programs are recommended to other users facing these data acquisition and entry problems. Listings may be found in Appendix C.

The Toluene Aerosol Experiments

The long-prepared series of photochemical aerosol experiments was conducted from June 20 to October 2, 1985. Toluene was the aromatic hydocarbon reactant. Oxides of nitrogen were added to all runs, using a ratio of NO : NO_2 of approximately 3 : 1. The air was kept "weakly" humidified by bubbling the main air fill stream through an unheated cannister of water during the several hour filling period; relative humidities near 50 % were measured in the room temperature sample line from the bag. Ammonium sulfate seed aerosol of approximately 0.04μm number average diameter, after drying, was generated by a continuous flow atomizer in few minutes time, for those runs in which seed aerosol was included. Seed aerosol concentrations were to be varied over a range where suppression of nucleation could (hopefully) be observed.

The first ten runs were conducted in simple single chamber mode, allowing the batch reaction to proceed in a nearly 65 cubic meter volume. Two concentrations of initial chemical reactants were chosen as targets. The low concentration runs were to have 1.2 ppm toluene, 0.45 ppm NO and 0.15 ppm NO_2; the high concentration runs were to have 3.6 ppm toluene, 1.35 ppm NO, and 0.45 ppm NO_2, for a fixed starting molecular hydrocarbon to NO_x ratio of 2 : 1. Initial concentrations were not exactly on target owing to the slight differences in the filling rates and plumpness of the bag at the start of the run. Equipment difficulties often delayed the start of the runs, necessitating the addition of more dilution air to replace that being sampled and that leaking out of the no-longer-new teflon bag. (Under no conditions was air added once the run had actually started.) In each case, the chamber concentrations were allowed to stabilize (indicating adequate mixing) before the opaque blue tarp was removed from the run, allowing the photochemistry to begin.

Table 1 summarizes the single mode experiments. In the very first experiment we succeeded in fully suppressing homogeneous nucleation, and in the second, with one-tenth the initial aerosol concentration, a large amount of homogeneous nucleation occurred. Thus we had discovered the seed aerosol concentration range of interest for the low hydrocarbon and NO_x case. Further experiments demonstrated the effect of outdoor variables on the aerosol system, as we could not control the number of particles that would form for low or zero aerosol concentrations to better than a factor of two. (Intermittent cloudiness, as occurred during run 20, understandably may have slowed the photochemistry which leads to aerosol production.) Mass conversion yields of toluene to aerosol (assumed to be near unit density) were typically from two to five percent.

The growth of the aerosol though various size channels of the EAA and OPC was also clearly observable. Often a channel would peak at two different times, once as the seed aerosol moved through that particle size interval, and again as the homogeneously nucleated aerosol passed through that size. Generally the two EAAs, the laser OPC, and the two CNCs were in fairly good agreement as to particle number.

The last three experiments of the single mode runs (37, 39, and 41) were high reactant concentration runs. Counterintuitively, the high concentration runs were slower (as it took longer for NO to disappear), and, although much higher aerosol mass loadings did result, the number of particles nucleated was essentially the same as for the low reactant concentration runs. This came as a surprise and is not yet understood.

As a result of the single chamber runs, it was clear that the addition of seed aerosol could not only suppress homogeneous nucleation, but could do it so efficiently that the addition of seed aerosol actually lowers the resulting total particle number concentrations appreciably. The effect of day to day random variables was appreciable, making quantitative conclusions difficult.

The photochemical aerosol experiments were continued using a dual chamber mode in which the teflon bag was divided in half by means of a PVC pipe resting on aluminum conduit. (Actually, the PVC pipe was tied down on both ends, to assure a reasonably good air seal between the two halves of the bag, despite the tendency of the PVC and aluminum support to bend unevenly.)

Table 2 summarizes the results of the dual chamber experiments. In five of the seven runs, the gas phase concentrations began the same in both sides. In four of those five runs, seed aerosol was only introduced to one chamber. The gas phase chemistry was essentially identical throughout the run for both sides, as expected. (What differences there were in gas phase concentrations and aerosol mass can perhaps be attributed to small differences in initial concentrations or more likely to slightly increased solar radiation by reflection, both of which would favor the north side, A, closer to the roof lab.)

The dual chamber runs allowed a more quantitative assessment of the suppression of nucleation by aerosol, since conditions should be the same for both sides, except for the aerosol concentration. In run 45, where all concentrations were initially equal between the two sides, the southern side (B) showed about a fifteen percent higher peak aerosol number concentration. The actual reason for this discrepancy is not known, but it gives us an estimate of the consistency of behavior between the two sides. (Gas phase concentrations differed by only a couple of percent between the sides.)

In the later dual chamber runs, a molecular ratio of 4 : 1 for hydrocarbon to NO_x was used to obtain a higher generation rate of condensable vapors. This was necessary, since the peak particle number concentrations resulting from the experiments went down as the surface to volume ratio went up (a problem with the dual runs, especially when the teflon is getting old and has many tiny leaks), and as solar radiation diminished with the end of summer. Nevertheless, the dual runs provided valuable information about the suppression of nucleation by seed aerosol. In run 57, about 3500 cm^{-3} particles were able to entirely suppress homogeneous nucleation that otherwise would have generated eight times that many particles. And in run 53, 2000 cm^{-3} suppressed about 70% of a nucleation burst that would have produced 30,000 cm^{-3}. Thus, the resulting number of particles in this photochemical system could be reduced by up to an order of magnitude by the addition of small seed aerosol. This is in good agreement with the predictions made by the SNM model, back in Chapters 4 and 6.

Conclusions

A complete validation of the SNM model was not possible for the toluene photochemical system, because the properties (and even the identity, at present) of the condensable species are not known. Thus surface tension (or dimensionless surface) energy and vapor pressure (or dimensionless source rate) are unknown parameters, and there is insufficient data to even estimate them independently, even assuming that the SNM model is correct. The varying outdoor temperatures, and subsequent influence on vapor pressure, and thus saturation ratio and nucleation rate, also makes modeling difficult. The suppression of nucleation by seed aerosol is a phenomenon that is fairly independent of the physical property values, according to the SNM model simulations, which is why this phemomenon could be confirmed in the absence of accurate physical property assumptions.

From the point of view of atmospheric chemistry, these experiments have revealed several interesting points. A few percent of the toluene photoxidation products have equilibrium vapor pressures well below one ppm and may form aerosols. The aerosol formation would be by condensation rather than homogeneous nucleation because atmospheric aerosol concentrations are sufficiently high to suppress nucleation of products resulting from reasonable ambient toluene concentrations (which are less than what were used for the smog chamber experiments). Increased generation of condensable vapor does not necessarily seem to mean that more new particles will be formed, contrary to expectations.

A more detailed analysis of the aerosol data from the toluene experiments is currently underway, which is unfortunately beyond the scope of this thesis. Temperature is clearly a very important variable, as a few degrees change in temperature makes a noticeable effect on the vapor pressure and hence the saturation ratio and nucleation rate.

References

Crump, J.G., Flagan, R.C., and Seinfeld, J.H. (1983). Particle wall loss rates in vessels. *J. Aerosol Sci.* 12:405–415.

Leone, J.A. (1984). *Ph.D. thesis entitled* "Studies in photochemical smog chemistry: I. Atmospheric chemistry of toluene II. Analysis of chemical reaction mechanisms for photochemical smog."

Leone, J.A., Flagan, R.C., Grosjean, D., and Seinfeld, J.H. (1985). An outdoor smog chamber and modeling study of toluene–NO_x photooxidation. *Int. J. Chem. Kinet.* 17:177-216.

McMurry, P.H., and Rader, D.J. (1985). Aerosol wall losses in electrically charged chambers. *Aerosol Sci. Technol.* 4:249–268.

Table 1: Summary of 1985 Single Chamber Toluene Runs

Run	Date	T_{max}	N_i	N_{max}	Tol_i	Δ Tol	V_{max}	Yield	Nucl
		°C	K/cc	K/cc	ppm	ppm	$\mu g\, m^{-3}$	%	
16	20-Jun	40	5.0	–	0.96	0.31	27.	2.3 %	None
18	27-Jun	44	0.4	12.1	1.30	0.50	64.	3.4 %	Much
20	29-Jun	41	5.4	–	1.45	0.42	60.	3.8 %	None
22	03-Jul	46	3.5	5.5	1.25	0.40	120.	7.9 %	Slight
26	12-Jul	40	0.	6.5	0.79	0.36	45.	3.3 %	Full
31	22-Jul	38	6.0	–	1.25	0.46	55.	3.1 %	Slight
35	26-Jul	42	0.	14.0	1.08	0.64	65.	2.7 %	Full
37	29-Jul	38	0.	12.0	3.80	1.80	330.	4.8 %	Full
39	31-Jul	38	5.5	–	4.30	1.70	250.	3.9 %	None
41	05-Aug	41	2.0	2.5	3.10	1.30	120.	2.4 %	Slight

Table 2: Summary of 1985 Dual Chamber Toluene Runs

Run	Date	T_{max}	N_i	N_{max}	Tol_i	Δ Tol	V_{max}	Yield	Nucl
		°C	K/cc	K/cc	ppm	ppm	$\mu g\,m^{-3}$	%	
43A	28-Aug	49	0.	6.0	4.90	1.30	260.	5.3 %	Full
43B			0.	4.0	3.90	0.80	127.	4.2 %	Full
45A	30-Aug	53	2.7	4.7	1.18	0.51	27.	1.4 %	Some
45B			2.7	5.5	1.18	0.51	28.	1.4 %	Some
48A	13-Sep	44	2.7	–	1.45	0.47	22.	1.2 %	None
48B			0.	10.7	1.45	0.54	21.	1.0 %	Full
53A	20-Sep	42	1.8	10.4	2.71	0.82	65.	2.1 %	Some
53B			0.	32.5	2.71	0.77	61.	2.1 %	Full
55A	23-Sep	45	8.7	–	2.65	0.95	40.	1.1 %	None
55B			0.	27.9	2.74	0.88	36.	1.1 %	Full
57A	25-Sep	41	3.7	–	2.95	0.89	56.	1.7 %	None
57B			0.	28.4	2.83	0.85	50.	1.5 %	Full
60A	02-Oct	44	1.5	8.8	3.00	1.01	55.	1.4 %	Some
60B			3.0	–	1.42	0.51	4.	0.2 %	None

CHAPTER 8:

CONCLUSIONS

Two major models have been developed here for simulating the evolution of an aerosol. The first, ESMAP, is a size-sectionalized, multicomponent computer model which includes the major physical processes (coagulation, condensation, nucleation, and deposition) which affect the size distribution of a spatially uniform aerosol. This model was used to predict the behavior of a system with a vapor source which hence might undergo homogeneous nucleation and rapid condensational growth. From these simulations arose a second and simpler model known as the SNM, which follows only the saturation ratio (S), total aerosol number (N), and total aerosol mass (M). Conceptually and computationally, the SNM model was more efficient at describing the nucleation and growth processes of interest. The two models are used to understand the dynamic interplay between the processes of nucleation and condensational growth, allowing us to find the dimensionless parameters that determine whether many small particles or a few large particles will result from a burst of nucleation.

From numerous simulations, we have found that nucleation and growth dynamics in the absence of seed aerosol are governed by the dimensionless rate of generation of condensable vapor. When the dimensionless source rate is low, meaning vapor molecules are generated slowly relative to the rate at which they collide with each other, smaller supersaturations and smaller aerosol number concentrations are achieved during the burst of nucleation. In the absence of seed aerosol, the dimensionless final number concentration (the aerosol number concentration divided by the saturated vapor concentration, which removes the vapor pressure dependence) is found to go as the dimensionless source rate raised to approximately the 1.4 power when nucleated particles grow into the continuum transport regime. For dimensionless source rates significantly higher than unity, the nucleation rate will lag behind classical nucleation theory because the monomer concentration is changing too rapidly for the critical cluster concentration to remain in steady state with it. At still higher saturation ratios and dimensionless source rates, the assumption of classical nucleation breaks down more seriously as cluster-cluster collisions enhance the nucleation rate. A system driven by a steady vapor source will be relatively insensitive to the assumed nucleation function. On the other hand, a system driven by a high initial supersaturation but with no subsequent vapor source will be highly sensitive to the assumed nucleation expression, and would be more appropriate for comparing different nucleation theories. The system driven by an initial supersaturation shows dynamics which are dominated by a burst of nucleation occurring at essentially initial conditions, and a good analytic approximation for its behavior during the burst of nucleation has been derived.

When seed particles are present, the system behavior depends primarily on the relative seed aerosol concentration, which is defined as the ratio of initial aerosol number concentration to the number concentration which would have been produced by homogeneous nucleation alone, i.e., in the absence of seed aerosol. When the relative seed aerosol concentration is low (less than about 0.001), the resulting number concentration is unaffected by the seed aerosol. When the relative seed aerosol concentration is high (greater than about 100), homogeneous nucleation is mostly or entirely suppressed. For relative seed aerosol concentrations of order 0.1 to one, slight suppression of homogeneous nucleation is predicted for the initially supersaturated system, while substantial or complete suppression of nucleation is predicted for the steady vapor source system. The different behavior for the two types of nucleating systems is due to the vapor build-up phase of the steady vapor source system. During the vapor build-up, seed particles have a chance to deplete the vapor and to grow larger before the saturation ratio is high enough for significant homogeneous nucleation to occur. The seed aerosol may remove vapor faster than it is produced and prevent homogeneous nucleation entirely, which can not happen when the initial saturation ratio is high enough to cause nucleation virtually instantaneously. Larger seed particles and higher supersaturations allow for a greater observable suppression of homogeneous nucleation by the seed aerosol.

Partial verification of the SNM model predictions was found in the two experimental systems analyzed. In our own experiments, the complete suppression of homogeneous nucleation by fairly low concentrations of aerosol has been shown for a toluene–NO_x photochemical system. Relative initial number concentrations of about 0.1 or higher suppress all observable nucleation, and partial suppression of homogeneous nucleation is observed with somewhat lower seed aerosol concentrations, as expected. These smog chamber experiments suggest that from one to five percent of the mass of toluene which photooxidizes has a very low vapor pressure and will condense on surrounding aerosol or homogeneously nucleate. Using a computerized data acquisition system, our instrumentation allowed us to follow the smooth growth of seed aerosols by many thousands of times in mass, leading to a relatively monodisperse condensation aerosol. Since the surface tension and vapor pressure of the condensing vapor or vapors are not known (although the SNM model allows us to estimate them, but not independently), the SNM model could not be used a priori to predict the number of particles which would result, in the absence of seed aerosol.

The SNM model was also used to analyze experiments conducted in a physiochemically well-characterized system with a high initial supersaturation but no subsequent vapor source. The SNM model described the behavior of the system to within experimental uncertainties after one key assumption was made. The condensing species, dibutylphthalate, homogeneously nucleated much faster than predicted by classical homogeneous nucleation theory, yet much slower than predicted by the rival Lothe-Pound nucleation theory. Hence an intermediate nucleation expression was used, which was found to be consistent with several previous nucleation studies on this compound. The SNM model with enhanced nucleation gave good agreement with experiment over several orders of magnitude of resulting number concentrations. Additionally, the observed slight suppression of nucleation by seed aerosol agreed with SNM predictions, although a system with a pulse source of vapor is not a desirable one for observing suppression of nucleation.

CHAPTER 9:

SUGGESTIONS FOR FUTURE WORK

As a result of the modeling and experiments described in this work, several possibilities for new or continued effort arise.

In the modeling realm, the computer codes used do not simulate fully and accurately all the physical processes which may apply to a nucleating and growing aerosol. None of the codes considers unsteady state homogeneous nucleation, where the collisions of subcritical clusters appreciably accelerate the nucleation rate. (A paper not yet in print by Wu et al., using a version of ESMAP modified to include the discrete treatment of cluster populations, promises to fill this gap.) There is a more modest case of unsteady state nucleation (which, by a consideration of time constants, should apply to some of the situations in this thesis where the SNM model was used) in which cluster-cluster collisions are negligible, yet the nucleation rate lags behind the classical rate because the time constant for establishment of steady state cluster populations is significant compared to the time constant for change in the monomer concentration (due either to a high dimensionless source rate or high initial supersaturation and thus high condensation rates); in such a case, fewer particles would be formed than predicted by steady state homogeneous nucleation.

The SNM model does not provide a full aerosol size distribution, and fixed-size sectional models such as ESMAP suffer from numerical diffusion which artificially broadens the size distribution. Adding the method of characteristics for condensational growth to the SNM treatment of the vapor concentration and of the homogeneous nucleation process should give accurate size distributions when coagulation is negligible, and could lead to more accurate prediction of resulting number than the SNM model.

In the experimental realm, further work using aromatic hydrocarbons and NO_x and seed aerosol is being conducted. Implementation of dual sampling lines for separate aerosol and chemical sampling promises to improve the quality of gas phase data. More careful attention to temperature and starting times of experiments should improve comparability of different runs, and exclusive operation in the dual chamber mode will also facilitate comparisons and the rate at which useful data can be collected.

Although improved technique in smog chamber aerosol experiments should provide more quantitative results, the outdoor smog chamber will still have its limitations. It is useful for reproducing atmospheric phenomena, controlled by solar radiation and diurnal temperature variation. It is not suited for a precise investigation of basic physical behavior, such as the SNM and ESMAP models try to predict, because of the uncontrolled parameters which accompany experiments under ambient conditions outdoors. Unless physical properties such as surface tension and vapor pressure can be obtained for the condensable species, good model validation is not possible. Even in well-characterized systems, such as dibutylphthalate in the PSM, the relationship between nucleation rate and saturation ratio is difficult to measure accurately.

APPENDIX A:

ESMAP PROGRAM DOCUMENTATION AND LISTING

*** ESMAP DOCUMENTATION: SUMMARY ***

This is a preliminary users' guide to the Expanded Sectional Multicomponent Aerosol Package (ESMAP), completed by Dale R. Warren in March 1984. This program grew out of the Multicomponent Aerosol (MAEROS) Code written by Fred Gelbard. Both were written at Caltech under the guidance of John H. Seinfeld, and deal with modeling the evolution of a spatially uniform aerosol. The new package handles homogeneous nucleation and condensation coupled to a varying gas-phase concentration, with improved methods for treating condensational growth and deposition, while retaining the full sectionalized treatment of coagulation. The new code is written in a structured language, FORTRAN-77, and has been run on VAX 11/780's. (Note: if program is to be run on a different machine, with a different FORTRAN, it may be necessary to make revisions to the code.)

The aerosol size distribution is represented in discrete form as an array of sectional masses, Q(I). In addition to MS size classes, the aerosol may consist of KC different components. The multicomponent sectional model places the mass of component K and size range L into section Q(I), where I=K+(L-1)*KC. Two extra elements are allocated, Q(MS*KC+1) for the condensible vapor mass concentration, and Q(MS*KC+2) for the mass of freshly nucleated particles.

ESMAP will optionally include (or ignore) a wide variety of physical processes: coagulation, condensation, evaporation, homogeneous nucleation and deposition. Control flags allow the user much flexibility in modeling cases of interest, often permitting the use of alternate expressions to evaluate a given rate. (Some of these are useful for comparing alternate theories, and others are just left over from the debugging phase.) The code is written in a modular form, so subroutines can readily be modified. A large quantity of information is stored in COMMON blocks, available as needed by the subroutines.

ESMAP proceeds in two major steps. The first step is the calculation of the sectional coefficients for coagulation, condensation, and deposition. The coagulation coefficients require a substantial effort, requiring evaluation of several similar types of double integrals. The condensation coefficients must be based on a fixed saturation ratio (which may be a variable for ESMAP), and thus must be scaled later to the actual saturation ratio, as well as later corrected for the Kelvin effect. The sectional coefficients may be stored in a data file for future use or inspection, so it is not always necessary to perform the first step. The second step is the time integration of the sectionalized differential equations governing the mass distribution, Q. Currently time integration is performed by the EPISODE package, a Gear method, modified to handle the approach of sections to zero mass.

The primary output of ESMAP consists of a printout (to a *.OUT file) of the multi-component sectional mass distribution at selected times, as well as saturation ratio, total number and surface area, as well as a few other parameters. Optionally ESMAP will also generate a record of the size distribution (*.DIS) that may readily be used as input for plotting, a record of the dimensionless condensation and nucleation parameters (*.DIM) also suitable for plotting, or an extended printout of the nucleation parameters (*.NUC) with time. Other occasionally informative files that may be generated consist of warnings from the EPISODE integration routine (*.EPI), a summary of negative mass produced by the inaccuracy of the time integration (*.NEG), and an optional step-by-step debugging of the nucleation routine (*.DJ).

!	ESMAP is available as a set of FORTRAN subroutines, each in
!a separate file (.FOR), plus a set of labeled COMMON segments and
!parameters (in .INC files) which are automatically included into
!the FORTRAN subroutines. (If your compiler doesn't accept the
!INCLUDE statement, the .INC files may readily be included with
!a text editor.) There are also command files to compile (COMP.COM)
!the subroutines, collect (MLIB.COM) them into a library (MCALIB.OLB),
!and to run (ESMAP.COM) the sample program, which includes the
!MA.FOR main program. (The compiled program optionally allows the
!user to input some key parameters at run time, and ESMAP.COM causes
!this input to be read from ESMAP.INP, which, along with APDATA.INC,
!CHOOSE.INC and MA.FOR, defines the parameters for a simulation.)

June 1985 Revision: To increase transportability of the code, ESMAP is being distributed as a single FORTRAN file with all subroutines and COMMON included, so the above paragraph may be ignored. However, ESMAP.COM still causes some key parameters to be read in from the ESMAP.INP file.

The results of a standard test case (including initial aerosol, condensation, nucleation, and coagulation) are kept in CIT2.OUT, with sectional coefficients saved in CIT2.CO. When the code is run on another installation, the files TEST2.OUT and TEST2.CO should be produced, which can be compared with the previously mentioned files. Also included on tape is CIT2NC.OUT, an identical run except coagulation is omitted (DOCOAG is .FALSE.), revealing not only the effect of coagulation on such a system, but also verifying that the condensation algorithm conserves total number (only the burst of homogeneous nucleation affects the number concentration then). The test case shows that the process of homogeneous nucleation is partially quenched by the initial aerosol. (A higher total number concentration would result if the simulation proceeded with no initial aerosol.)

Descriptions of the SUBROUTINES, the VARIABLES, and the COMMON blocks contained in ESMAP are available in other .DOC files, following.

*** ESMAP DOCUMENTATION: SUBROUTINES ***

This is a brief guide to the subroutines used in the Expanded Sectional Multicomponent Aerosol Package (ESMAP), written by Dale R. Warren in 1984. All of the following subroutines are available in a .FOR file of the same name, on disk or tape. (DRIVE is found in EPIS.FOR) Also see related documentation on Usage, Variables, and Common Blocks for ESMAP.

ASKFOR(MS,DELDEP,RATEC,RELE,ABSE,MF,KTOL,BNAME,CNAME,SNAME,ASKME,BATCH)

CALLED BY:	MAIN PROGRAM (initialization phase)
CALLS:	NONE
PURPOSE:	To enable the user to modify the listed simulation parameters at run time.

BETA(Y,X,TGAS,PGAS,NBTYPE) FUNCTION

CALLED BY:	BETCAL, GAUS2, CHECKE
CALLS:	RHODD
PURPOSE:	To calculate the Coagulation Rate.

BETCAL(X,RELER,ABSER,ROUND,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE) FUNCTIO

CALLED BY:	COEF, GAUSBT
CALLS:	GAUS2, BETA
PURPOSE:	To calculate the Inner Integral of the Sectional Coagulation Coefficients.

CALCON(QT,QVAP,SR,CONRAT,Z)

CALLED BY:	DIFFUN, MAEROS
CALLS:	SRATIO
PURPOSE:	To calculate the total rate of condensation (excluding the Kelvin effect) and the saturation ratio and condensation coefficient scaling parameter.

CALSIZ(DPMIN,DPMAX)

CALLED BY:	MAIN PROGRAM (initialization phase)
CALLS:	None.
PURPOSE:	To set /SIZES/ and /XSIZES/ COMMON for sectional particle sizes. (Used for geometric spacing of sections.)

CHECKE(TIME,DELTIM,Q,TGAS,PGAS,IPRNT,IFLAG,NEWCOF)

CALLED BY:	MAEROS
CALLS:	BETA
PURPOSE:	To confirm the input values to MAEROS are reasonable, and otherwise abort program with error message.

COEF(NEWCOF,TGAS,PGAS,IPRNT)

CALLED BY:	MAEROS
CALLS:	GAUSBT, GAUS2, SETGAS (Also ERRORO)
PURPOSE:	To calculate the Sectional Coefficients.

DEPOST(X,DUMMY,TGAS,PGAS,NBTYPE) FUNCTION

CALLED BY:	COEF, GAUS2
CALLS:	OLDDEP, RHODD (Also COTH and DEBYE1)
PURPOSE:	To calculate the deposition rate coefficient.

DIFFUN(NEQ,T,Q,DQDT)

CALLED BY:	MAEROS

CALLS: CALCON, JMKS

PURPOSE: To calculate the time derivative of the sectional mass array, Q.

DRIVE(NEQ,TIME,HO,Q,TOUT,AERROR,KTOL,MF,IFLAG)

CALLED BY: MAEROS

CALLS: (Other Subroutines in the EPISODE Package)

PURPOSE: To integrate a set of simultaneous O.D.E.s.

GAUS2(F,XL,XU,RELER,ABSER,ROUND,ANSWR,IER,EXTRA1,EXTRA2,EXTRA2,NEXTRA)

CALLED BY: BETCAL, COEF

CALLS: BETA

PURPOSE: To calculate the integral of an explicit function using repeated quadrature.

GAUSBT(F,XL,XU,RELER,ABSER,ROUND,ANSWR,IER,IPRNT,FIXSZ,BASEZ,INNER, TGAS,PGAS,NBTYPE)

CALLED BY: COEF

CALLS: BETCAL

PURPOSE: To calculate the Inner Integral of the sectional coagulation coefficient expression.

GROWTH(X,DUMMY,TGAS,PGAS,NBTYPE)

CALLED BY: COEF, GAUS2

CALLS: RHODD (Also FDFS or FDCE)

PURPOSE: To calculate the condensational growth rate coefficient of a particle.

J(S,RJN,CRATE)

CALLED BY: JMKS

CALLS: (Internal Subroutines)

PURPOSE: To calculate the homogeneous nucleation rate.

JMKS(S,RJM,CONRAT,GC)

CALLED BY: DIFFUN

CALLS: J

PURPOSE: To interface the cgs-units nucleation routine with the MKS-units MAEROS package.

MAEROS(TIME,DELTIM,Q,TGAS,PGAS,IPRNT,IFLAG,NEWCOF)

CALLED BY: MAIN

CALLS: COEF, CHECKE, SETGAS, CALCON, DIFFUN, DRIVE (Integrator)

PURPOSE: Driver for the time evolution of an aerosol.

MAIN PROGRAM

CALLED BY: None

CALLS: MAEROS, ASKFOR, CALSIZ, SETGAS, PRINFO, STORE, PUTCOF, PRESET, JSET, NLIST, PREPLT, PRINTO, SAVEP, PRSTAT

PURPOSE: To initialize the variables and call the MAEROS driver, and print out a record of the time evolution of the aerosol.

NLIST(IO,IARG)

CALLED BY: MAIN

CALLS: None

PURPOSE: To list the physical parameters of a condensing system.

OLDDEP(X,DUMMY,TGAS,PGA,NBTYPE) FUNCTION

CALLED BY: DEPOST

CALLS: RHODD

PURPOSE: To calculate a deposition rate coefficient using boundary layer theory.

PEDERV(N,T,Y,PD,NO)

CALLED BY: EPISODE integration package

CALLS: (None)

PURPOSE: To calulate the Jacobian for the mass array Q. Not implemented, except as a dummy subroutine.

PREPLT(FNAME)

CALLED BY: MAIN PROGRAM (initialization phase)

CALLS: None

PURPOSE: Save header information for the .DIM file containing dimensionless condensation and nucleation parameters.

PRESET(TEMP,PRES,RATEG)

CALLED BY: MAIN PROGRAM (initialization phase)

CALLS: None

PURPOSE: To initialize the cgs nucleation routine COMMON /NUCLO/ from the MKS COMMON in /CONDNS/, /GAS/, and /STOKES/.

PRINFO(IP,METHOD)

CALLED BY: MAIN PROGRAM (initialization phase)

CALLS: None

PURPOSE: To save a record of the integration package parameters used.

PRINTO(Q,TIME,VOLU,IFLAG,IPRNT)

CALLED BY: MAIN PROGRAM (initialization phase)

CALLS: None

PURPOSE: To print out a complete description of the aerosol size distribution at a given time.

PRSTAT(IPRNT)

CALLED BY: MAIN PROGRAM

CALLS: None

PURPOSE: To print out how much effort has been required of the time integration routine.

PUTCOF(ITP)

CALLED BY: MAIN

CALLS: None

PURPOSE: To store just-calculated sectional coefficients in the /DBLK/ COMMON, to enable future interpolation.

RHODD(V,D,RHO)

CALLED BY: BETA, CALSIZ, COEF, DEPOST, GROWTH, OLDDEP

CALLS: None

PURPOSE: To interconvert particle mass and diameter.

SAVEP(TIME,Q)

CALLED BY: MAIN PROGRAM

CALLS: None

PURPOSE: To save dimensionless condensation & nucleation profiles in a *.DIM data file.

SETGAS(TGAS,PGAS)

CALLED BY: BETA, COEF, DEPOST, MAIN, MAEROS, OLDDEP

CALLS: None

PURPOSE: Calculate gas properties and save in /GAS/ COMMON.

SRATIO(QVAP) FUNCTION

CALLED BY: CALCON, PRINTO, SAVEP

CALLS: None

PURPOSE: To calculate the saturation ratio from the vapor mass concentration.

SSKELV(SR,DP,DIKELV) FUNCTION

```
  CALLED BY:    DIFFUN

  CALLS:        None

  PURPOSE:      To calculate the effective supersaturation (driving force)
                at the surface of a spherical droplet.

STORE(IODIR,NEWCOF,TGAS,PGAS,IPRNT,CNAME)

  CALLED BY:    MAIN PROGRAM (initialization phase)

  CALLS:        None

  PURPOSE:      To store or restore sectional coefficients to or from
                a data file.
```

```
       ***     ESMAP   DOCUMENTATION:  COMMON Block Usage      ***

************************************************************************

       Use of COMMON in the ESMAP Code by Dale Warren
       All COMMON Blocks and Variables Are Described

************************************************************************

------------------------------------------------------------------------

  AVGCOF.INC:            ! Sectional Coefficients

 COMMON /AVGCOF/ COEFAV(NCMAX)

       COEFAV  Sectional Coefficients for Coagulation, Deposition, Growth
               Calculated as sectional integrals prior to time integration

       Usage:  COEFAV consists of mean coefficients integrated over
               each section prior to time integration.  The COEFAV may
               be interpolated over temperature and pressure from
```

```
            COMMON /DBLK/ sectional coefficients, but otherwise
            COEFAV are constant through the time integration,
            and are calculated automatically by MAEROS.

        Note:  To see how the coefficients are ordered, see /INDEX/.

-----------------------------------------------------------------------


COMMON /DBLK/ CT1P1(NCMAX),CT1P2(NCMAX),CT2P1(NCMAX),CT2P2(NCMAX)

        CT1P1   Sectional Coefficients calculated for TGAS1,PGAS1
        CT1P2   Sectional Coefficients calculated for TGAS1,PGAS2
        CT2P1   Sectional Coefficients calculated for TGAS2,PGAS1
        CT2P2   Sectional Coefficients calculated for TGAS2,PGAS2

        Usage:  Coefficients are calculated by MAEROS prior to
                time integration.  If only one temperature and
                pressure are of interest (as specified by NEWCOF),
                only CT1P1 is calculated.  Otherwise COEFAV is
                obtained by a linear interpolation in temperature
                and pressure from these arrays.

-----------------------------------------------------------------------

  DEPSIT.INC            ! Deposited Masses

COMMON /DEPSIT/ DEPSIT(3,K)

        DEPSIT  Deposited Mass onto each of 3 surfaces, by each of
                 KC components [kg]

        Usage:  Deposited mass is updated with each return from
                MAEROS.  Values only used for printout.

        Note:   DEPSIT values are approximate, by differencing mean rates.

-----------------------------------------------------------------------

  FLAGS.INC             ! Control Flags

COMMON /CFLAGS/ DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2,DOLIMT,DODVAP,
                GEOSEC,DONCON,NOEVAP

        DOINIT  LOGICAL flag to enable initial aerosol mass
        DOSORC  LOGICAL flag to enable (sectional) aerosol source terms
        DODEPO  LOGICAL flag to enable deposition process
        DOCOAG  LOGICAL flag to enable coagulation process
        DOCOND  LOGICAL flag to enable condensation process
        DOCON2  LOGICAL flag to enable second-order method for condensation
        DOLIMT  LOGICAL flag to limit sectional decay times (used with DOCON2)
        DODVAP  LOGICAL flag to solve vapor concentration as an ODE
        GEOSEC  LOGICAL flag to use geometrically spaced sections (DEL fixed)
        DONCON  LOGICAL flag to enable number-conserving (1st-Ord) condensatio

        NOEVAP  LOGICAL flag to disable (net) evaporation

                Note: If DOCOAG is true, the geometric constraint applies,
                which requires each section to span a particle mass range
                whose upper limit is at least twice its lower limit; thus
                no more than 9.97 sections per decade in particle diameter
                may be used.

COMMON /NFLAGS/ DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE

        DOKELV  LOGICAL flag to include Kelvin Effect in condensation process
        DONUCL  LOGICAL flag to enable homogeneous nucleation
        DOCLBL  LOGICAL flag to calculate nucleation by cluster balance model
        DOSCAV  LOGICAL flag to include cluster-aerosol scavenging (if DOCLBL)
        LESSDI  LOGICAL flag to increase cluster balance N2/N1 ratio (DOCLBL)
        USEBCE  LOGICAL flag to use Chapmann-Enskog (a la Pete McMurry) rather
                    than modified Fuchs-Sutugin for transitional condensation

                Note: DOCLBL, DOSCAV, and LESSDI (if true) introduce
                      modifications of classical nucleation theory that
                      are available only in the long, experimental
                      J subroutine, but not in the JCLASS file.

COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS

        DEBUGJ  LOGICAL flag to debug homogeneous nucleation routine
        SAVNUC  LOGICAL flag to save time record of nucleation parameters
                    in BNAME.NUC file (large)
        SAVDIM  LOGICAL flag to save time record of dimensionless
                    condensation & nucleation parameters in BNAME.DIM file
        SAVDIS  LOGICAL flag to save time record of size distribution
                    in BNAME.DIS file

COMMON /VFLAGS/ NUFLAG,TCON,RATEG

        NUFLAG  Nucleation Method Flag (only used if DOCLBL is true)
        TCON    Minimum Time in which section can be depleted (if DOLIMT)
        RATEG   Rate of Generation of Condensible Vapor [kg/cu.m/sec]


        Usage:  Flags are set prior to any calculations to control
                which processes are included, which algorithms are
                used, and what parameters are stored for the
                time simulation.  Normally flags should not be changed
                without resetting MAEROS package and perhaps
                recalculation sectional coefficients.

------------------------------------------------------------------------

  GAS.INC
```

COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS

TEMP	Gas Temperature [K]
PRES	Gas Total Pressure [Pa]
PSAT	Saturation Vapor Pressure [Pa]
DENAIR	Air (Background Gas) Density [kg/cu.m]
FREEMP	Air (Background Gas) Free Mean Path [m]
VISCOS	Air (Background Gas) Viscosity [kg/m/sec]

Usage: Gas properties are saved in this COMMON block. They will be updated only if temperature or pressure of the gas changes with time, and are handled automatically with each call to MAEROS.

INDEX.INC: ! Indices or Pointers Used With Sectional Coefficients

COMMON /INDEX/ MS,KC, NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF

MS	Number of Size Sections
KC	Number of Chemical Components
NB2A	Offset for Coagulation Coefficients of Type 2A
NB2B	Offset for Coagulation Coefficients of Type 2B
NB3	Offset for Coagulation Coefficients of Type 3
NB4	Offset for Coagulation Coefficients of Type 4
NDEPST	Offset for Deposition Coefficients
NGROW	Offset for Condensational Growth Coefficients
ICONDN	Condensation Flag: 1 for Condensation, 0 for None
NUMCOF	Total Number of Coefficients

Usage: Pointers to the sectional coefficient array are determined initially and saved in this COMMON. MS and KC are specified by main program, while other variables are set by MAEROS and should not be modified.

Note: The offset must be added to the index of the coefficient to get the array index in COEFAV, the array of sectional coefficents.

PARINT.INC ! Time Integration Parameters

COMMON /PARINT/ RELE,ABSE,KTOL,MFEPI,H0

RELE	Relative Error Tolerance for Q in time integration
ABSE	Absolute Error Tolerance for Q in time integration
KTOL	Flag for Type of Error Tolerances to use
MFEPI	Time Integration Method Flag
H0	Current Local Time Step for Integration Package

```
   Usage:  Parameters are initially set to determine error
           tolerance and method for integration package.
           HO is revised intermittently by the integration
           package, while other parameters are normally fixed.

   Note:   This COMMON block is set up for use with a
           modified version of the LSODE integration package.

-------------------------------------------------------------------------

  PHYSPT.INC:

COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME

       ACELOV  Ceiling Surface Area to Container Volume Ratio [/m]
       AFLROV  Floor Surface Area to Container Volume Ratio [/m]
       AWALOV  Vertical Wall Surface Area to Container Volume Ratio [/m]
       VOLUME  Total Container Volume [cu.m]

               Note:  Surface to Volume Ratios are all used only for positive
                       DELDEP.  In the unified treatment (DELDEP<0.),
                       only their sum matters, and only if there is
                       Deposition.  VOLUME is important in printout,
                       but does not effect intensive properties.

COMMON /WALLS/  DELDEP,TURBDS,AKE

       DELDEP  Thickness of Boundary Layer [m], or Deposition Flag:
               -1. for spherical container, -2. for cube, <=-4. for none.
       TURBDS  Turbulent Energy Dissipation Rate [J/cu.m./sec]
       AKE     Turbulence Parameter (Ke in lit.) [/sec]
               Note TURBDS and AKE are used only for DELDEP<0.

COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE

       DELSAT  Reference Supersaturation, Sref-1.
       CONMW   Molecular Weight of Condensing Species
       GASMW   Molecular Weight of Background Gas (e.g., air)
       SURTEN  Surface Tension (of condensible) [Nt/m]
       DIFFUS  Diffusivity (of condensible vapor) [m*m/sec]
       BCE     Dimensionless Group, DIFFUS/(VEL*FREEMN)

               Note if RATEG<0. then DELSAT is actual supersaturation,
               not just reference.  If DODVAP is false, RATEG is
               also the (fixed, total) rate of condensation.

COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA

       DENSTY  Condensed Phase Density [kg/cu.m.]
       CHI     Particle Dynamic Shape Factor [-]
       FSLIP   Particle Slip Coefficient [-]
       STICK   Particle Sticking Coefficient [-]
       GAMMA   Agglomeration Shape Factor [-]
```

```
COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP

        FTHERM  Thermophoretic Parameter [-]
        TGRADC  Temperature Gradient to Ceiling [K/m]
        TGRADF  Temperature Gradient to Floor [K/m]
        TGRADW  Temperature Gradient to Walls [K/m]
        TKGOP   Thermal Conductivity Ratio, Gas to Particle [-]


        Usage:  Most of the physical parameters of the simulation
                are contained in these blocks of COMMON.  They
                must be set before any call to MAEROS, and changing
                them will usually necessitate recalculating the
                sectional coefficients.

------------------------------------------------------------------------

  PSRATE.INC            ! Sectional Aerosol Source Rates

COMMON /PSRATE/  PSRATE(NEMAX)

        PSRATE  Source Rate of Sectional Aerosol [kg/cu.m/sec]


        Usage:  PSRATE is set before any CALL to MAEROS.

        Note:   SRATE is ordered like the Q array.

------------------------------------------------------------------------

  ROUND.INC             ! Computer Dependent

COMMON /ROUND/ UROUND

        UROUND  Unit Round-Off Error (largest X that 1.+X=1.)
                        5.96E-8 for VAX REAL*4

        Usage:  Main program must initialize this correctly for
                best integration accuracy.

------------------------------------------------------------------------

  SIZES.INC             ! Sectional Diameters and Masses

COMMON /SIZES/ DS(MMAX1),VS(MMAX1)

        DS      Array of Sectional Particle Diameters [m]
        VS      Array of Sectional Particle Masses [kg]

        Usage:  Main Program must initialize these correctly,
                as by a call to CALSIZ (if GEOSEC).  Must
                recalculate sectional coefficients if changed.
```

```
------------------------------------------------------------------

  TPSET.INC:             ! (T,P) Set For Interpolation of Sectional Coefficient


COMMON /TPSET/ TGAS1,TGAS2,PGAS1,PGAS2

       TGAS1   Minimum Temperature for Interpolation
       TGAS2   Maximum Temperature for Interpolation
       PGAS1   Minimum Pressure for Interpolation
       PGAS2   Maximum Pressure for Interpolation

       Usage:  Initialized by main program; not normally modified
               as sectional coefficients would need recalculation.

------------------------------------------------------------------

  XSIZES.INC             ! Sectional Particle Sizes Expressed in Log(mass)

COMMON /XSIZES/ XS(MMAX1),DEL(MMAX)

       XS      Array of Sectional Particle Size by Log Mass [log10(kg)]
       DEL     Array of Sectional Particle Size Intervals [log10(kg)]

       Usage:  Initialized by main program to correspond to /SIZES/
               COMMON values.  Not normally modified.

       Note:          DEL(I) = XS(I+1) - XS(I)

------------------------------------------------------------------

       Special Nucleation (J) Subroutine COMMON (generally cgs units):

COMMON /NUCL0/
       T       Gas Temperature [K]
       VP      Vapor Pressure [dynes/sq.cm]
       MW      Molecular Weight of Condensible [g/gmole]
       DENSTY  Density of Condensible Liquid [g/cc]
       SURTEN  Surface Tension of Condensible Liquid [dynes/cm]
       SRATE   Generation Rate of Condensible Vapor [ug/cu.m/sec]

COMMON /NUCL1/
       SUE     Dimensionless Surface Energy (=SURTEN*SAM/BK/T)
       RSCALE  Characteristic Collision Rate (=CS*CS*VELQ*SAM) [#/sec/cc]
       TB      Characteristic Collision Time, Sat. (=CS/RSCALE) [sec]
       TS      Characteristic Source Time, Sat. (=CS*WEIGHT/SRATE) [sec]
       DIMSOR  Dimensionless Source Rate (=TB/TS)
       WEIGHT  Mass [ug/cu.m] per Number Concentration [#/cc]

COMMON /NUCL2/
       VL      Liquid Molar Volume [cc/mole]
       VM      Monomer Volume (Liquid) [cc/molecule]
```

```
        DIAM     Monomer Diameter (Liquid Extrapolation) [cm]
        SAM      Monomer Surface Area (Liquid Extrapolation) [sq.cm]
        CS       Saturated Monomer Concentration (Vapor) [#/cc]
        VELQ     1/4 Mean Monomer Velocity (Vapor) [cm/sec]
        VPAT     Monomer Vapor Pressure (atm)
        DIKELV   Characteristic Kelvin Diameter (=4*SURTEN*VM/BK/T) [cm]
        DSMIN    (Lowest) Sectional Aerosol Diameter Boundary [cm]

COMMON /NUCL3/
        SR       Saturation Ratio
        GCRIT    Critical Number (Monomers per Critical Nucleus)
        DIMAA    Dimensionless Equivalent Surface Area (re: Sat Monomers)
        BETAS    Reference Collision Frequency (mono/mono) [#/sec]
        NFLAG    Flag for Type of Cluster Balance Expression (0,1,2,3)
        TN       Estimated Delay Time for Steady State Nucleation
        RMNU     Mass Rate of Nucleation [ug/cu.m/sec at size DSMIN]
        RMNMIN   Minimum Non-Negligible Mass Nucleation Rate [ug/cu.m/sec]
```

*** ESMAP DOCUMENTATION: VARIABLES ***

This is a brief guide to the variables used in the Expanded Sectional Multicomponent Aerosol Package (ESMAP), completed by Dale R. Warren at Caltech in 1984. See other .DOC files for further information on ESMAP.

Except as noted, the standard Fortran Convention is used for data types (I-N INTEGER, others REAL). A "*" designation denotes a physical parameter that the user must specify for his system of interest, while a "+" denotes a parameter relating to how the system is solved computationally, which the user may have cause to modify.

```
ABSE    /PARINT/     + Absolute Error Tolerance for Time Integration Scheme
ABSER                  Absolute Error Tolerance for Sectional Integrals
ACELOV  /CHAMBR/     * Ceiling Surface Area / Chamber Volume [/m]
AFLROV  /CHAMBR/     * Floor Surface Area / Chamber Volume [/m]
AKE     /WALLS/      * Turbulence Parameter (Ke) for Deposition [/sec]
AKN                    Knudsen Number, Mean Free Path / Particle Radius [-]
AN      PARAMETER      Avogadro's Number [molecules/mole]
ASK                    LOGICAL flag to enable printing of questions
ASKME                + LOGICAL flag to input selected parameters at run time
AWALOV  /CHAMBR/     * (Vertical) Wall Surface Area / Chamber Volume [/m]
BASESZ                 Size for Sectional Integral Evaluation [kg ; log(kg)]
BATCH                + LOGICAL flag to supress queries (with ASKME true)
BCE     /CONDNS/     * DIFFUS/(VEL*FREEMN) [dimensionless group]
BK      PARAMETER      Boltzmann Constant [erg/K]
BMOBLX                 Particle Mobility
BNAME                  CHAR*16 Base FileName (sans .ext) for Simulation
```

CHI	/STOKES/	*	Particle Dynamic Shape Parameter
CNAME		+	CHAR*20 FileName containing existing Coefficients
COEFAV(I)	/AVGCOF/		Array of Sectional Coefficients at TGAS,PGAS
COLEFF			Collision Efficiency
CONMW	/CONDNS/	*	Molecular Weight of Condensible Species [g/mole]
CONRAT			Total Condensation Rate (ignore Kelvin) [kg/cu.m/sec]
CT1P1(I)	/DBLK/		Array of Sectional Coefficients at TGAS1,PGAS1
CT2P1(I)	/DBLK/		Array of Sectional Coefficients at TGAS2,PGAS1
CT1P2(I)	/DBLK/		Array of Sectional Coefficients at TGAS1,PGAS2
CT2P2(I)	/DBLK/		Array of Sectional Coefficients at TGAS2,PGAS2
D			Particle Diameter [m]
DEBUGJ	/SFLAGS/	+	LOGICAL flag to debug nucleation routine J (to *.DJ)
DEL(M)	/XSIZES/		Array of Sectional Widths, in X units [log(kg)]
DELDEP	/WALLS/	*	Deposition Boundary Layer Thickness [m], or flag
DELSAT	/CONDNS/	+	Reference Supersaturation for Sectional Coefficients
DELTIM			Step Size in Time for Next Printout [sec]
DENAIR			Density of Background Gas [kg/cu.m]
DENSTY	/STOKES/		Particulate (liquid or solid) Density [kg/cu.m]
DEPSIT(3,K)	/DEPSIT/		Array of Deposition onto 3 Surfaces by Comp K [kg]
DIFFUS	/CONDNS/	*	Diffusivity of Condensible [m*m/sec]
DIN			Minimum Aerosol Diameter [m]
DOCLBL	/NFLAGS/	*	LOGICAL flag to use cluster-balance nucleation theory
DOCOAG	/CFLAGS/	*	LOGICAL flag to enable coagulation process
DOCOND	/CFLAGS/	*	LOGICAL flag to enable condensation process
DOCON2	/CFLAGS/	*	LOGICAL flag to use 2nd-order condensation expression
DODEPO	/CFLAGS/	*	LOGICAL flag to enable deposition process
DODVAP	/CFLAGS/	*	LOGICAL flag to solve vapor as coupled ODE
DOINIT	/CFLAGS/	*	LOGICAL flag to enable initial aerosol mass
DOKELV	/NFLAGS/	*	LOGICAL flag to include Kelvin Effect on condensation
DOLIMT	/CFLAGS/	*	LOGICAL flag to limit 2nd-order condensational terms
DONCON	/CFLAGS/	*	LOGICAL flag to make condensation conserve number
DONUCL	/NFLAGS/	*	LOGICAL flag to enable homogeneous nucleation process
DOSCAV	/NFLAGS/	*	LOGICAL flag to include scavenging in Cl-Bal Nucleatio
DOSORC	/CFLAGS/	*	LOGICAL flag to enable aerosol sources (SRATE)
DQVAP			Time Derivative of Vapor Concentation QVAP [kg/cu.m./s
DS(M1)	/SIZE/		Array of Sectional Particle Diameters (Boundaries) [m]
DUMMY			Unused (or temporary) REAL Variable
F			Factor relating generalized flux / continuum flux [-]
FIXSZ			Size for Sectional Integral Evaluation [kg ; log(kg)]
FREEMN			Monomer Mean Free Path [m]
FREEMP			Gas Mean Free Path (for particle motion) [m]
FSLIP	/STOKES/	*	Particle Slip Coefficient [-]
FTHERM	/THERM/	*	Thermophoretic Parameter [-]
GAMMA	/STOKES/	*	Agglomeration Shape Factor [-]
GASMW	/CONDNS/	*	Molecular Weight of Background Gas (often air) [g/mole
GEOSEC	/CFLAGS/	+	LOGICAL flag to use geometrically-spaced sections
HO	/PARINT/		Current Local Step Size for Integration Routine
ICONDN	/INDEX/		Flag to calculate condensation coefs (0=No,1=Yes)
IFLAG			Flag for Subroutines (use varies)
INNER			Flag (0,1,2) for type of inner sectional integral

IPRNT		+ Logical Unit Number for Output Device (often 6)
IWORK()		Integer Workspace for Integration Routine
K		Loop Index Relating to Kth Component
KC	/INDEX/	* Number of Chemical Components
KTOL	/PARINT/	Flag for Type of Error Tolerances to Use In Integratio
LESSDI	/NFLAGS/	* LOGICAL flag to adjust Cl-Bal to increase N1/N2 ratio
MF		Method Flag (see MFEPI)
MFEPI	/PARINT/	Method Flag for Integration Routine (EPISODE)
MKMAX	PARAMETER	Array Dimension, Maximum Value for NQMK
MMAX	PARAMETER	+ Array Dimension, Maximum Value for MS
MMAX1	PARAMETER	Array Dimension, Equal to MMAX+1, for size boundaries
MS	/INDEX/	+ Number of Size Sections
NB2A	/INDEX/	Offset (in COEFAV) for Type 2A Coagulation Coefficient
NB2B	/INDEX/	Offset (in COEFAV) for Type 2B Coagulation Coefficient
NB3	/INDEX/	Offset (in COEFAV) for Type 3 Coagulation Coefficients
NB4	/INDEX/	Offset (in COEFAV) for Type 4 Coagulation Coefficients
NBTYPE		Flag for type of coefficient
NCMAX	PARAMETER	Array Dimension, Maximum Value for NUMCOF
NDEPST	/INDEX/	Offset (in COEFAV) for Deposition Coefficients
NEMAX	PARAMETER	+ Maximum Number of ODE's for which code is dimensioned
NEWCOF	/INDEX/	+ Flag to control calculation of sectional coefficients
NGROW	/INDEX/	Offset (in COEFAV) for Condensational Coefficients
NOEVAP	/CFLAGS/	* LOGICAL flag to disable evaporation process
NQMK		Number of Aerosol Sections Used (=MS*KC)
NQN		Subscript (in Q) for Total Nucleation (=MS*KC+2)
NQV		Subscript (in Q) for Vapor Concentration (=MS*KC+1)
NRMAX	/EPCOMR/	Maximum Element of Q To Keep Non-Negative (for EPIS)
NRMIN	/EPCOMR/	Minimum Element of Q To Keep Non-Negative (for EPIS)
NUFLAG	/VFLAGS/	* Flag for type of cluster-balance nucleation (0-3)
NUMCOF	/INDEX/	Number of Sectional Coefficients (per set), <= 2M*M+4M
NWMAX	PARAMETER	Array Dimension, Work Space Size for ODE solver
ONE	PARAMETER	Unit Value (1.0) in REAL*4 form [-]
PGAS		* Total Gas Pressure [Pa=Nt/sq.m]
PGAS1	/TPSET/	* Minimum Total Pressure for Interpolations [Pa]
PGAS2	/TPSET/	* Maximum Total Pressure for Interpolations [Pa]
PRES	/GAS/	Gas Total Pressure [Pa]
PI	PARAMETER	Geometric Pi Value (3.1415927) in REAL form [-]
PSAT	/CONDNS/	* (Saturation) Vapor Pressure of Condensible [Pa]
Q(N)		Array of Sectional Masses (Augmented) [kg/cu.m]
		Note order : M1(K...), M2, ..., MS ; NQV, NQN
QT(M)		Array of Total Mass by Size Section [kg/cu.m]
QVAP		Total Condensible Mass in Vapor Phase [kg/cu.m]
RATEG	/VFLAGS/	* Rate of Generation of Condensible Species [kg/cu.m/sec
RELE	/PARINT/	+ Relative Error Tolerance for Time Integration Scheme
RELER		Relative Error Tolerance for Sectional Integral [-]
RGAS	PARAMETER	Universal Gas Constant, MKS units (8314.4) [J/K/kgmole
S		Saturation Ratio [-]
SAVDIM	/SFLAGS/	* LOGICAL flag to save dimensionless parms (to *.DIM)

SAVDIS	/SFLAGS/	* LOGICAL flag to save size distribution (to *.DIS)
SAVNUC	/SFLAGS/	* LOGICAL flag to save nucleation summary (to *.NUC)
SNAME		+ CHAR*20 FileName for Saving Coefficients
SR	/NUCL3/	Saturation Ratio [-]
SRATE(N)	/SRATE/	* Array of Sectional Aerosol Source Rates [kg/cu.m/sec]
SS		Effective Supersaturation (Driving Force), Surface [-]
STICK	/STOKES/	* Particle Sticking Coefficient [-]
SURTEN	/CONDNS/	* Surface Tension of Condensible [Nt/m]
SUM		A Variable Used for Summation [-]
TCON	/VFLAGS/	* Sectional Disappearance Time Constant (if DOLIMT) [sec
TEMP	/GAS/	Gas Temperature [K]
TGAS		* Gas Temperature [K]
TGAS1	/TPSET/	* Minimum Temperature for Interpolations [K]
TGAS2	/TPSET/	* Maximum Temperature for Interpolations [K]
TGRADC	/THERM/	* Temperature Gradient to Ceiling [K/m]
TGRADF	/THERM/	* Temperature Gradient to Floors [K/m]
TGRADW	/THERM/	* Temperature Gradient to Wall [K/m]
TIME		Current Time in Simulation [sec]
TKGOP	/THERM/	* Thermal Conductivity Ratio, Gas/Particle [-]
TOUT()		* Array of Output Times (for Print Out) [sec]
TURBDS	/WALLS/	* Turbulent Energy Dissipation Rate [m**2/sec**3]
U		Particle Mass as size parameter (see Y) [kg]
UROUND	/ROUND/	+ Unit Round-Off Error (largest that UROUND+1.=1.) [-]
USEBCE	/NFLAGS/	* LOGICAL flag to use Chapmann-Enskog not mod. Fuchs-Sut
V		Particle Mass as size parameter [kg] (see X)
VEL		Mean Kinetic Molecular Velocity [m/sec]
VISCOS		Background Gas (air) Viscosity [kg/m/sec]
VTERM		Gravitational Terminal Velocity [m/sec]
VTHRML		Thermal Deposition Velocity [m/sec]
VOLUME	/CHAMBR/	* Chamber Volume [cu.m]
VS(M1)	/SIZES/	Array of Sectional Particle Masses (boundaries) [kg]
X		Log of Particle Mass as size parameter [log(kg)]
XS(M1)	/XSIZES/	Array of Sectional Sizes, Log10(mass) [log(kg)]
Y		Log of Particle Mass as size parameter [log(kg)]
Z		Condensation Scaling Factor, (SR-1.)/DELSAT [-]
ZERO	PARAMETER	Zero Value (0.0) in REAL form [-]

```
      PROGRAM MA           ! Main Multi-Component Sectional Aerosol Program
C
C*****************************************************************************
C  Uses the Extended Sectional MultiComponent Aerosol Package (ESMAP)  *
C        Models the Time Evolution of a MultiComponent Aerosol          *
C               For Documentation see ESMAP.DOC                          *
C*****************************************************************************
C    Programmed by Dale R. Warren for use on Caltech Systems (1984)    *
C Incorporates Homogeneous Nucleation & Number-Conserving Condensation *
C               Written in VAX-77 Structured FORTRAN                     *
C*****************************************************************************
C  Based on MAEROS, ISSUED BY SANDIA LABORATORIES (FRED GELBARD, 1982) *
C*****************************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )       ! MKS
      PARAMETER ( NEMAX = 218 )           ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                 ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )       ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )         ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )               ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      DIMENSION Q(NEMAX)          ! Major Dependent Variable Array (Masses)
      DIMENSION TOUT(0:30)        ! Array of Output Times
      CHARACTER*20 FNAME,CNAME,SNAME     ! FileNames: Output and Coefficient
      CHARACTER*16 BNAME                 ! Basic FileName (sans .ext)
      LOGICAL BATCH,KNOWCO,ASKME,GEOTIM ! Local Logic
C       Choose Control Flags    CHOOSE.INC
C
C       COMMON for Control Flags        AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
C
C       Generally the Control Flags are .TRUE. unless something
C        is being omitted from the model.
C
      DATA DOINIT /.TRUE./       ! Include Initial Mass distribution?
      DATA DOSORC /.FALSE./      ! Include Particle Source Rate terms?
      DATA DODEPO /.FALSE./      ! Include Deposition?
      DATA DOCOAG /.TRUE./       ! Includes Coagulation?
      DATA DOCOND /.FALSE./      ! Includes Condensation?
      DATA DOCON2 /.FALSE./      ! Use 2nd order intersectional condensation?
      DATA DOLIMT /.TRUE./       ! Limit intersectional flux to avoid negative?
      DATA DODVAP /.FALSE./      ! Handle Condensible Vapor with ODE?
      DATA GEOSEC /.FALSE./      ! D(L+1)/D(L) is same for all L?
      DATA DONCON /.TRUE./       ! Use Number-conserving Condensation Algorithm

      DATA NOEVAP /.TRUE./       ! May Ignore Possibility of Evaporation?
C
      DATA DOKELV /.FALSE./      ! Includes the Kelvin Effect on Condensation?
      DATA DONUCL /.FALSE./      ! Includes Homogeneous Nucleation?
      DATA DOCLBL /.FALSE./      ! Use Cluster Balance Equations for Nucleation

      DATA DOSCAV /.FALSE./      ! Includes Cluster Scavenging for Nucleation?
      DATA LESSDI /.FALSE./      ! Try to 'correct' C.B.E. to near Classical?
      DATA USEBCE /.FALSE./      ! Use Chapmann-Enskog instead of M.F.S. Cond.?
      DATA DEBUGJ /.FALSE./      ! Write Stepwise Nucleation Record to FOR011?
      DATA SAVNUC /.FALSE./      ! Write Nucleation Parameters to FOR020?
      DATA SAVDIM /.FALSE./      ! Write Dimensionless Parameters to FOR098?
      DATA SAVDIS /.TRUE./       ! Write Mass Distribution (Q array) to FOR026?
C
      DATA NUFLAG / 0 / ! selects form for nucleation cluster balance (0,1,2,3

      DATA TCON   / 5.0 /        ! Time Constant in Seconds for DOLIMT
      DATA RATEG / 0. / ! 0.01 ug/cu.m./sec of condensible generated
C
C        Most of the flags are self-explanatory.  About the others:
C
C        DOCON2 acts to reduce numerical diffusion with the intersectional
C                  condensation term by letting boundary flux out of section I
C                  depend on section I+1 as well as I.
C        If DOCON2, there may be numerical problems caused as section I
C                  vanishes.  DOLIMT (with an appropriate TCON) steps
C                  down the order of disappearance to depend only on the
C                  mass in section I itself, when trouble appears.
C        DONCON overrides DOCON2 and DOLIMT, using a first-order algoritm
C                  that properly conserves number concentration by condensation.
C        DOCLBL uses a cluster population balance (exact form selected by
C                  NUFLAG) instead of the pure classical thermodynamic form.
C                  DOSCAV requires DOCLBL.
C        PHYSPT.INC to establish uniform COMMON for physical properties
C        COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)          ! Sectional Diam & Masses
      COMMON /TPSET/ TGAS1,TGAS2,PGAS1,PGAS2     ! T,P set for interpolation
      COMMON /PSRATE/  PSRATE(NEMAX)     ! Sectional Particle Source Rates
      COMMON /DEPSIT/ DEPSIT(3,2)        ! Deposited Masses
C  DEPSIT array is 3 surfaces by KCOMP components.  Approximate values.
      COMMON /ROUND/ UROUND      ! Unit Round-Off Error (5.96E-8 for VAX REAL*4

      COMMON /PARINT/ RELE,ABSE,KTOL,MFEPI,HO    ! Integration Parameters
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
C
      COMMON /INDEX/ MS,KC,NOV,NON       ! Sectional Pointers
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT      ! Nucleation COMMON
      COMMON /EPCOMY/ YMIN,HMAXMX                      ! COMMON for EPIS
C
      DATA UROUND / 5.961E-8 /    ! Set for the Caltech 11-780 VAXes
      DATA FNAME,CNAME,SNAME / 'MCA.OUT','AER.CO','NEW.CO' /
C
      DATA RELE / 0.001 /         ! Allow 0.1% Local Error
      DATA ABSE / 1.E-20 /        ! Accurate to 1.E-11 ug/cu.m. (default)
      DATA KTOL / 8 /             ! Control Relative Error to YMIN, reject <-YMI

      DATA MFEPI / 20 /           ! For Stiff Systems, Avoids Finding Jacobian
      DATA HO / 1.E-2 /           ! Initial Time Step for Integration
C
C       This is APDATA.INC -- Aerosol Property Data for Test Case 3
C
C***             Set /CHAMBR/ values:    for 100 cu. m bag
C
      DATA ACELOV        / 0.4 /           ! Ceiling Surface:Volume Ratio [/m]
      DATA AFLROV        / 0.4 /           ! Floor Surface:Volume Ratio [/m]
      DATA AWALOV        / 0.55 /          ! Wall Surface:Volume Ration [/m]
      DATA VOLUME        / 1. /  ! Chamber Volume [cu.m]
C
C***             Set /WALLS/ values:
C
      DATA DELDEP        / -4. /           ! Flag supresses deposition
      DATA TURBDS        / 0.001 /         ! Turbulent Energy Loss Rate [J/cu.m/s

      DATA AKE           / 0.1 /           ! Turbulence Parameter (Ke) [/sec]
C
C  In ESMAP, DELDEP (the boundary layer thickness of MAEROS) is
C    ordinarily negative and used as a flag:
C       -1. ==> Sphere with AKE turbulence parameter (re Jim Crump's paper)
C       -2. ==> Cube with AKE turbulence parameter
C       -4. or less ==> No Deposition
C       positive ==> Old boundary layer model used, thickness DELDEP [m]
C
C
C***             Set /CONDNS/ values:
C
      DATA DELSAT        / 1.0 /           ! Reference Supersaturation for COEFAV
      DATA CONMW         / 100. /          ! Molecular Weight of Condensible
      DATA GASMW         / 29.0 /          ! Molecular Weight of Air
      DATA SURTEN        / 25.E-3 /        ! Surface Tension, typical organic
      DATA DIFFUS        / 0. /
C      DATA DIFFUS        / 0.0430E-4 /    ! Diffusivity [m*m/sec] (old: 0.0411)
      DATA BCE           / 0. /
C      DATA BCE           / 0.3333333 /    ! Dimensionless # (SURTEN/FREEMN/VEL)
C      DATA BCE           / 1.17 /         ! Dimensionless # if BCE theory used
C       Diffusivity now set for MW=100., Density 1 g/cc by BS&L D(AB) formula
C        based on kinetic theory for hard spheres of unequal sizes
```

```
C       BCE was for Pete McMurry's organic, not ours.
C       BCE is irrelevant if modified Fuchs-Sutugin theory used.
C       Note: DELSAT must be correct if RATEG<0.; for positive RATEG
C               DELSAT is a reference supersaturation for calculation
C               of the sectional condensation coefficients a priori.
C
C
C***            Set /STOKES/ values:
C
      DATA DENSTY       / 10.5E3 /       ! Liquid Density [kg/cu.m]
      DATA CHI          / 1. /           ! Particle Dynamic Shape Factor
      DATA FSLIP        / 1.37 /         ! Particle Slip Coefficient
      DATA STICK        / 1. /           ! Particle Sticking Coefficient
      DATA GAMMA        / 1. /           ! Agglomeration Shape Factor
C
C***            Set /THERM/ values:      ! Needed Only for Thermophoresis
C
      DATA FTHERM       / 1. /           ! Thermophoresis Parameter
      DATA TGRADC       / 0. /           ! Temp. Gradient to Ceiling [K/m]
      DATA TGRADF       / 0. /           ! Temp. Gradient to Floor [K/m]
      DATA TGRADW       / 0. /           ! Temp. Gradient to Walls [K/m]
      DATA TKGOP        / 0.05 /         ! Gas/Particle Thermal Conductivities
C
C***            Set /GAS/ PSAT (assumes only one temperature used)
C
      DATA PSAT / 1.E-5 / ! Our Medium Vapor Pressure Standard Organic [Pa]
C  1.E-5 Psat corresponds to 1.E-4 dynes/sq.cm or 9.87E-11 atm or 7.5E-8 torr
C      DATA PSAT        / 1.E-6 / ! Our Low Vapor Pressure Standard Organic
C
      DATA QVAP / 0. /           ! No Initial Vapor (SR=0.)
      DATA TGAS1,TGAS2,PGAS1,PGAS2 / 298.,450.,1.0133E5,7.09E5 /
C
      DATA MS,KC / 3,1 /         ! 36 size sections by 2 components
      DATA IPRNT / 1 /           ! Set to print to file # 1
      DATA KNOWCO / .TRUE. /     ! Flag TRUE if COEFAV taken from file
      DATA BATCH  / .TRUE. /     ! Flag TRUE if no interactive I/O
      DATA ASKME  / .TRUE. /     ! Flag TRUE if user asked for parameters
      DATA GEOTIM / .FALSE./     ! Flag TRUE if geometric spacing of output tim

C
      LOGICAL CTEST /.TRUE./
C
C***            BEGIN BY ALLOWING REVISION OF PARAMETERS
C
      CALL ASKFOR(MS,DELDEP,RATEC,RELE,ABSE,MFEPI,KTOL,
     $              BNAME,CNAME,SNAME,ASKME,BATCH)
      RATEG=RATEC        ! Change variables in COMMON
      YMIN=ABSE
      IF (CTEST) GEOTIM=.TRUE.
C
C***            OPEN DATA FILES
C
      IF (LENCH(BNAME).GT.0 .AND. BNAME.NE.'N') THEN
```

```
        IF (DEBUGJ) THEN
          FNAME=BNAME//'.DJ'
          OPEN (UNIT=11,FILE=FNAME,STATUS='NEW')
        ENDIF
        IF (SAVNUC) THEN
          FNAME=BNAME//'.NUC'
          OPEN (UNIT=20,FILE=FNAME,STATUS='NEW')
        ENDIF
        IF (SAVDIM) THEN
          FNAME=BNAME//'.DIM'
          OPEN (UNIT=98,FILE=FNAME,STATUS='NEW')
        ENDIF
        IF (SAVDIS) THEN
          FNAME=BNAME//'.DIS'
          OPEN (UNIT=26,FILE=FNAME,STATUS='NEW')
        ENDIF
        FNAME=BNAME//'.EPI'
        OPEN (UNIT=3,FILE=FNAME,STATUS='NEW')
        FNAME=BNAME//'.NEG'
        OPEN (UNIT=4,FILE=FNAME,STATUS='NEW')
        FNAME=BNAME//'.WAR'
        OPEN (UNIT=13,FILE=FNAME,STATUS='NEW')
        FNAME=BNAME//'.OUT'             ! This is the Output File Name
      END IF
C
      IF (IPRNT.NE.6) THEN
        OPEN (UNIT=IPRNT,FILE=FNAME,STATUS='NEW')
      ENDIF
C
      IF (CNAME.EQ.'N') KNOWCO=.FALSE.
C
      IF (SAVDIS) WRITE(26,26) MS,KC    ! For size distribution record
   26 FORMAT(1X,'MS=',I5,4X,'KC=',I5)
C
C***            CALCULATE SECTIONAL PARTICLE SIZE RANGES
C
      DPMIN=5.6E-9                ! Smallest diameter in meters
      DPMAX=32.0E-9               ! Largest diameter in meters
      CALL CALSIZ(DPMIN,DPMAX)    ! Calculate DS,VS,XS,DEL sectional size arrays
      DIN=DPMIN                   ! Nucleation of particles into smallest size
C
C***            INITIALIZE SECTIONAL MASSES TO ZERO
C
      NQMK=MS*KC                  ! Number of Aerosol Sections by component,size
      NQV=NQMK+1                  ! Q Subscript for Vapor Mass Concentration
      MQN=NQMK+2                  ! Q Subscript for Nucleated Mass (to DIN)
      DO I=1,NQN                  ! Initialize All Sections
        Q(I)=ZERO                 ! Initialize to No Mass
        PSRATE(I)=ZERO            ! Initialize to No Source Rate
      END DO
C
C***            SET INITIAL SECTIONAL MASS DISTRIBUTION
C
```

```
C      MDIV=MS/9                  ! 1/9 of sectional range has initial aerosol
C       1.E-14 supposedly standard ==> 27.65 /cc
C      QINIT=5.E-16               ! Total mass concentration of initial aerosol
C      IF (CTEST.AND.DOINIT) THEN
C        MDIV=0                   ! USED ONLY FOR GROWTH TEST
C        Q(1)=1.E-15              ! 1.E-6 ug/cu.m grows for one million growth
C      END IF
C      IF (DOINIT) THEN           ! Allow initial concentration profile
C        DO I=MDIV+1,2*MDIV
C          Q(I*KC-KC+1) = QINIT / MDIV  ! Step function mass density distrib.
C        END DO
C      ENDIF
       Q(1)=1.E-10                        ! kg/cu.m (1.E-9 ug/cu.m.)
       Q(2)=1.E-10
       Q(3)=1.E-10
       Q(4)=1.E-10
       Q(5)=0.
       Q(6)=0.
C
C***              SET VARIOUS PARAMETERS
C
      TGAS=TGAS1          ! Use the sectional coefficients computed for
      PGAS=PGAS1          !  initial conditions TGAS1 and PGAS1
      CALL SETGAS(TGAS,PGAS)     ! Set /GAS/ Properties
C
      TIME=ZERO           ! Start at time zero
      INDEX=1             ! First Call to this Problem for DRIVES (Integrator)
      NEWCOF=2            ! Use TGAS1 and PGAS1 only
      SRI=1.0             ! Initial Saturation Ratio
      QVAP=SRI*CONMW*PSAT/(RGAS*TGAS)    ! Vapor Mass Concentration
      Q(NQV)=QVAP         ! Vapor Mass stored in augmented Q array
C
C***              PRINT MESSAGE ON INTEGRATION METHOD
C
      CALL PRINFO(IPRNT,'EPISODE ')
C      WRITE(IPRNT,49) MFEPI,RELE,KTOL,YMIN
C   49 FORMAT(/' USING MF=',I3,5X,'RELE=',1PE10.3,5X,
C     $ 'KTOL=',I2,5X,'YMIN=',E10.3/)
C
C***              HANDLE COEFFICIENT FILE(S)
C
      IF (KNOWCO) THEN
       IODIR=1.                   ! Flag to Get from File
       CALL STORE(IODIR,NEWCOF,TGAS,PGAS,IPRNT,CNAME)
       IF (IODIR.GE.0.) THEN              ! File matches
        NEWCOF=-IABS(NEWCOF)              ! Since know COEFAV already
        CALL PUTCOF(1)                    ! Save COEFAV in CT1P1
        WRITE(IPRNT,900) CNAME            ! Note source of COEFAV
  900   FORMAT(/' **** USING COEFFICIENTS FROM FILE ',A20,' ****'/)
       ELSE
        KNOWCO=.FALSE.                    ! Coefficient File Doesn't Match
       END IF
      END IF
```

```
C
C***             SET UP NUCLEATION COMMON AND PRINT OUT SUMMARY
C
      CALL PRESET(TGAS,PGAS,RATEG)       ! Set /NUCL0/ for J
      DINCM=100.*DIN              ! Smallest Section Diameter in cm
      CALL JSET(DINCM)            ! Set /NUCL1/, /NUCL2/, /NUCL3/ for J
      CALL TRSET                  ! Set DIFFUS and BCE in both COMMONs
      CALL NLIST(IPRNT,4)
      IF (IPRNT.NE.6) CALL NLIST(6,4)
      IF (BNAME.EQ.'N' .OR. LENCH(BNAME).EQ.0) BNAME=FNAME(1:16)
      CALL PREPLT(BNAME)
C
C***              SELECT OUTPUT TIMES (May scale to TS, or TB)
C
      IF (GEOTIM) THEN
        TOUT(0)=0.
        TMIN=0.1                          ! First Output Time (seconds)
        DO I=0,5                          ! Span Over Six Orders of Magnitude
          DO J=1,4                        ! x1, x2, x3, x5
            K=J
            IF (J.EQ.4) K=5
            TOUT(4*I+J)=TMIN*FLOAT(K)*10.**I
          END DO
        END DO
        NTIME=20
      ELSE       ! Use /NUCL1/ values of TB,TS,DIMSOR to select reasonable time

C        NTIME=12
C        IF (DIMSOR.LE.1.E-2) THEN
C          ITIME=1
C        ELSE IF (DIMSOR.LE.1.E0) THEN
C          ITIME=2
C        ELSE IF (DIMSOR.LE.5.) THEN
C          ITIME=5
C        ELSE IF (DIMSOR.LE.100.) THEN
C          ITIME=20
C        ELSE IF (DIMSOR.LE.1000.) THEN
C          ITIME=100
C        ELSE
C          STOP 'DIMSOR is too Large for Auto-Time Selection'
C        END IF
        ITIME=1
        NTIME=15
        IF (DIMSOR.GE.0.5) ITIME=2
        ITIME=1
        TS=100.                   ! 100 second time steps
        DO I=0,NTIME
          TOUT(I)=I*TS*ITIME      ! TS is characteristic source time scale
        END DO
      END IF
C
      HMAXMX=2.E-3*TOUT(1)        ! Maximum Episode Time Step Size (Seconds)
      HMAXMX=100.*TOUT(1)
      HMAXMX=50.                          ! Testing Try
C
C***              PRINT OUT INITIAL SIZE DISTRIBUTION
C
      IPFLAG=1
   80 CALL PRINTO(Q,TIME,VOLUME,IPFLAG,IPRNT)
      IF (SAVDIM) THEN
        CALL SAVEP(TIME,Q)       ! Save dimensionless values
      ENDIF
C
C***              DO TIME INTEGRATION OF SECTIONAL AEROSOL GROWTH
C
      STEPIO=1.0                 ! 0. < STEPIO <= 1.  Substep for Data Record
      IPFLAG=5
C
      DO ITIME=1,NTIME  ! Advance to each output time of interest
        DO SUBINT=STEPIO,1.,STEPIO       ! Subintervals for saving data
          TOUT1=(1.-SUBINT)*TOUT(ITIME-1)+SUBINT*TOUT(ITIME)
          DELTIM=TOUT1-TIME              ! Time Interval
          CALL MAEROS(TIME,DELTIM,Q,TGAS,PGAS,IPRNT,INDEX,NEWCOF)
          IF (SAVDIM) CALL SAVEP(TIME,Q)
        END DO
        IF (ITIME.EQ.1.AND..NOT.KNOWCO) THEN     ! Immediately create SNAME
          IODIR=0                 ! Flag set to Write Coefficients to File
          CALL STORE(IODIR,NEWCOF,TGAS,PGAS,IPRNT,SNAME)
          KNOWCO=.TRUE.
        ENDIF            ! ASCII Coefficient File has been saved ASAP
C
        CALL PRINTO(Q,TIME,VOLUME,IPFLAG,IPRNT)
        CALL PRSTAT(IPRNT)       ! Print Integration Statistics
      END DO
C
C***              DONE WITH CALCULATIONS AND PRINTOUT
C
      CLOSE (IPRNT)
      STOP 'MULTICOMPONENT AEROSOL (EPI) PROGRAM FINISHED'
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE ASKFOR(MS,DELDEP,RATEG,RELE,ABSE,MF,
     $  KTOL,BNAME,CNAME,SNAME,ASKME,BATCH)
C
C***********************************************************************
C
C  PURPOSE:
C       To allow specification of certain simulation parameters after
C        linking program.
C
C  ON ENTRY:
C       MS                Number of size sections
C       DELDEP            Deposition boundary layer or flag [m]
C       RATEG             Mass Condensation Rate (if fixed) [kg/cu.m/sec]
```

```
C       RELE               Relative Error Tolerance
C       ABSE               Absolute Error Tolerance
C       MF                 Method Flag for Gear integration routine, etc.
C       KTOL               Flag to select type of error tolerance (in EPISODE)
C       BNAME              Basic FILENAME of Run (sans .EXT)
C       CNAME              Existing Coefficient FILENAME ('N' for none)
C       SNAME              New Coefficient FILENAME (only if needed, 'N'= none)
C       ASKME              Local control flag to accept input (if TRUE)
C       BATCH              Local control flag to type prompts (if TRUE)
C
C  ON RETURN:
C       Variables may be set to new value.
C
C  COMMENTS:
C       Input will default to compiled value.
C
C***********************************************************************
C
      CHARACTER*16 BNAME
      CHARACTER*20 CNAME,SNAME
      LOGICAL*1 ASKME,BATCH,ASK
C
      IF (.NOT.ASKME) THEN
        TYPE 900
 900    FORMAT(/5X,'PROGRAM NOT USING PARAMETER FILE'/)
        RETURN
      END IF
C
      ASK=(.NOT.BATCH)            ! TRUE if Interactive Job
C
      IF (ASK) TYPE 110, MS
 110  FORMAT('$Enter MS (5-36 Sections) [',I3,'] : ')
      ACCEPT 203, IDUMMY
 202  FORMAT(I2)
 203  FORMAT(I3)
      IF (IDUMMY.GT.0) MS=IDUMMY
C
      IF (ASK) TYPE 120, DELDEP
 120  FORMAT('$Enter DELDEP (m) (-1. JGC or -9. NO DEP) [',
     $ 1PE10.3,'] : ')
      ACCEPT 315, DUMMY
 315  FORMAT(G15.7)
      IF (DUMMY.NE.0.) DELDEP=DUMMY
C
      IF (ASK) TYPE 130, RATEG
 130  FORMAT('$Enter RATEG (kg/sec/cu.m.) (-1. S.S.) [',
     $ 1PE10.3,'] : ')
      ACCEPT 315, DUMMY
      IF (DUMMY.NE.0.) RATEG=DUMMY
      IF (DUMMY.LT.-1.) RATEG=0.        ! Need Zeroing Option
C
      IF (ASK) TYPE 150, RELE
 150  FORMAT('$Enter RELE (relative error) [',1PE8.2,'] : ')
```

```
      ACCEPT 315, DUMMY
      IF (DUMMY.NE.0.) RELE=DUMMY
C
      IF (ASK) TYPE 155, ABSE
  155 FORMAT('$Enter ABSE (absolute error) [',1PE8.2,'] : ')
      ACCEPT 315, DUMMY
      IF (DUMMY.NE.0.) ABSE=DUMMY
C
      IF (ASK) TYPE 160, MF
  160 FORMAT('$Enter MFEPI (method flag) [',I2,'] : ')
      ACCEPT 202, IDUMMY
      IF (IDUMMY.NE.0) MF=IDUMMY
C
      IF (ASK) TYPE 165, KTOL
  165 FORMAT('$Enter KTOL for EPI (1-9) [',I1,'] : ')
      ACCEPT 202, IDUMMY
      IF (IDUMMY.NE.0) KTOL=IDUMMY
C
      IF (ASK) TYPE 800
  800 FORMAT('$Enter Identifying File Name : ')
      ACCEPT 400, BNAME
  400 FORMAT(A20)
      IF (BNAME.EQ.' ') BNAME='AEROSOL'          ! Default Output File
C
      IF (ASK) TYPE 810
  810 FORMAT('$Enter Coefficient Input File Name ? ')
  815 ACCEPT 400, CNAME
      IF (CNAME.EQ.'Y') GO TO 815        ! Ask again for Name
      IF (CNAME.EQ.' ') CNAME='AEROSOL.CO'      ! Default Input File
C
      IF (ASK) TYPE 820
  820 FORMAT('$Enter Coefficient Output File Name : ')
      ACCEPT 400, SNAME
      IF (SNAME.EQ.' ') SNAME='AEROSOL.CO'      ! Default Output File
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      FUNCTION BETA(Y,X,TGAS,PGAS,NBTYPE)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Coagulation Coefficient.
C       In addition to simple Brownian motion, gravity and
C        turbulence are included mechanisms, with additivity assumed.
C
C  ON ENTRY:
C       Y                 Log Mass of first particle [ln(kg)]
C       X                 Log Mass of second particle [ln(kg)]
C       TGAS              Gas Temperature [K]
```

```
C       PGAS              Gas Pressure, Total [Pa]
C       NBTYPE            Type of Coefficient Needed
C       /GAS/   DENAIR    Background Gas Density [kg/cu.m]
C       //      FREEMP    Background Gas Mean Free Path [m]
C       //      VISCOS    Background Gas Viscosity
C
C  ON RETURN:
C       BETA              Coagulation Coefficient
C
C  LOCAL VARIABLES:
C       V,U               Particle Masses (of X and Y) [kg]
C       DX,DY             Particle Diameters (of X and Y) [m]
C
C  COMMENTS:
C       Note BETA is a symmetric function in X and Y, BEFORE it is
C        sectionalized.  NBTYPE =  4,5 retain this symmetry.
C      REFERENCES:  FUCHS,N.A. 'MECHANICS OF AEROSOLS', 291-294,
C      PERGAMON (1964).  GIESEKE,J.A., LEE,K.W. AND REED,L.D.,
C      'HAARM-3 USERS MANUAL', BMI-NUREG-1991 (1978).  DRAKE,R.L.
C      'A GENERAL MATHEMATICAL SURVEY OF THE COAGULATION EQUATION,'
C      IN TOPICS IN CURRENT AEROSOL RESEARCH BY HIDY,G.M. AND
C      BROCK, J.R. (EDS.) VOL.3 PERGAMON, N.Y. 1972.
C
C***********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )       ! MKS
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS   ! Gas Properties
      U=EXP(Y)                           ! Mass of First Particle
      V=EXP(X)                           ! Mass of Second Particle
      DX=ZERO
      DY=ZERO
      CALL RHODD(V,DX,RHOX)              ! Calculate Particle Diameters
      CALL RHODD(U,DY,RHOY)
C
C***             AIR VISCOSITY, DENSITY, MEAN FREE PATH HELD IN /GAS/
C***              DOUBLECHECK TEMPERATURE & PRESSURE ARE CONSISTENT
C
      IF (TGAS.NE.TEMP.OR.PGAS.NE.PRES) THEN
        IF (TGAS.NE.TEMP) TYPE 21, TEMP,TGAS
21      FORMAT(/' WARNING: /GAS/ TEMP =',F7.1,' while TGAS=',F7.1 /)
        IF (PGAS.NE.PRES) TYPE 22, PRES,PGAS
22      FORMAT(/' WARNING: /GAS/ PRES =',1PE9.2,' while PGAS=',E9.2 /)
        CALL SETGAS(TGAS,PGAS)
      END IF
C
      AKX=2.*FREEMP/DX              ! Knudsen Number (X in air)
      AKY=2.*FREEMP/DY              ! Knudsen Number (Y in air)
      BMOBLX=1.+AKX*(FSLIP+.4*EXP(-1.1/AKX))
      BMOBLY=1.+AKY*(FSLIP+.4*EXP(-1.1/AKY))
C
C  CHI=DYNAMIC SHAPE FACTOR  ;  GAMMA=AGGLOMERATION SHAPE FACTOR
C
      FCHIX=CHI
      FCHIY=CHI
      FGAMX=GAMMA
      FGAMY=GAMMA
      DSUM=FGAMX*DX+FGAMY*DY
      VABDIF=.54444*ABS(RHOX*DX*DX*BMOBLX/FCHIX-RHOY*DY*DY*BMOBLY/FCHIY)
     $  /VISCOS
      DIFX=1.4642E-24*TGAS*BMOBLX/(DX*FCHIX*VISCOS)
      DIFY=1.4642E-24*TGAS*BMOBLY/(DY*FCHIY*VISCOS)
C
C        BROWNIAN COAGULATION COEFFICIENT
C
      VXSPED=SQRT(3.51E-23*TGAS/V)
      VYSPED=SQRT(3.51E-23*TGAS/U)
      VMEAN=SQRT(VXSPED*VXSPED+VYSPED*VYSPED)
      AMX=2.5465*DIFX/VXSPED
      AMY=2.5465*DIFY/VYSPED
      GX=((DX+AMX)**3-(DX*DX+AMX*AMX)**1.5)/(3.*DX*AMX)-DX
      GY=((DY+AMY)**3-(DY*DY+AMY*AMY)**1.5)/(3.*DY*AMY)-DY
      GMEAN=SQRT(GX*GX+GY*GY)
      BETA=DX+DY
      BETA=2.*PI*(DIFX+DIFY)*DSUM/(BETA/(BETA+2.*GMEAN) +
     $  8.*(DIFX+DIFY)/(VMEAN*BETA*STICK))
C
C***              ADD GRAVITATIONAL COAGULATION
C
      COLEFF=1.5*(AMIN1(DX,DY)/(DX+DY))**2
      BETA=BETA+.7854*STICK*DSUM*DSUM*VABDIF*COLEFF
C
C***              ADD TURBULENT COAGULATION
C
      TURB1=.1618*SQRT(TURBDS*DENAIR/VISCOS)*DSUM*DSUM*DSUM
      TURB2=.074*VABDIF*DSUM*DSUM*SQRT(SQRT(DENAIR*TURBDS*
     $  TURBDS*TURBDS/VISCOS))
      BETA=BETA+STICK*SQRT(TURB1*TURB1+TURB2*TURB2)
C
C***              INTERNAL CHECK FOR ERROR
C
      IF (BETA.EQ.ZERO) THEN
        TYPE 90, BETA,U,V,NBTYPE
 90     FORMAT(' BETA=',1PE10.3,5X,'U=',E10.3,5X,'V=',E10.3,
     $            5X,'NBTYPE=',I2)
        STOP 'BETA=0. SHOULD NOT HAVE OCCURRED'
      END IF
```

```
C
C***            CONVERT TO MASS SECTIONALIZED BETA
C***            THESE LINES MUST ALWAYS BE INCLUDED IN CODE,
C***             REGARDLESS OF THE FUNCTIONAL FORM OF BETA.
C
      GO TO (2,1,2,3,3,1),NBTYPE
    1 BETA=BETA/V
      RETURN
    2 BETA=BETA/U
      RETURN
    3 BETA=BETA*(U+V)/U/V        ! Note /(U*V) leads to divide by zero
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      FUNCTION BETCAL(X,RELER,ABSER,ROUND,IPRNT,FIXSZ,BASESZ,INNER,
     $                  TGAS,PGAS,NBTYPE)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Inner Integral of the Sectional Coagulation
C        Coefficients.
C
C  ON ENTRY:
C       X                 Outer Integral Size Value [log10(mass)]
C       RELER             Relative Error Tolerance for Sectional Integral
C       ABSER             Absolute Error Tolerance for Sectional Integral
C       ROUND             Unit Round-Off Error (largest X that 1.+X=1.)
C       IPRNT             Logical Unit Number for Output Device or File
C       FIXSZ             Size Limit for Inner Integral
C       BASESZ            Size Limit for Inner Integral
C       INNER             Flag (0,1,2) for Type of Sectional Coefficient:
C                          Inner Integral Has Following Range (where z=exp(x)):
C                                INNER=0 :      BASESZ to FIXSZ
C                                INNER=1 :      log(BASESZ-z) to FIXSZ
C                                INNER=2 :      FIXSZ to log(BASESZ-z)
C       TGAS              Gas Temperature [K]
C       PGAS              Gas Pressure [Pa]
C       NBTYPE            Type of Sectional Coefficient
C
C  ON RETURN:
C       BETCAL            Inner Integral
C
C  COMMENTS:
C       None.
C
C***********************************************************************
C
      EXTERNAL BETA
C
C***            USE INNER TO SET LIMITS ON INNER INTEGRAL
```

```
C
      IF (INNER.EQ.0) THEN
        YU=FIXSZ
        YL=BASESZ
      ELSE IF (INNER.EQ.1) THEN
        YU=FIXSZ
        YL=ALOG(BASESZ-EXP(X))
      ELSE
        YU=ALOG(BASESZ-EXP(X))
        YL=FIXSZ
      END IF
C
C***             Need Alternate Inner Integral Evaluation if Endpoints Converge
C
      IF (INNER.EQ.1) THEN
        ETEST=ABS(YU-YL)/(ABS(YU)+ABS(YL))
      END IF
C
      IF (INNER.EQ.1 .AND. ETEST.LT.500.*ROUND) THEN
C
C***             Use 2nd Order Taylor Expansion -DRW
C
        DELVL=EXP(X)/BASESZ
        YMEAN=0.5*(YU+YL)
        ANSWR=(DELVL+0.5*DELVL*DELVL)*BETA(YMEAN,X,TGAS,PGAS,NBTYPE)
      ELSE
C
        IER=1              ! YL & YU set properly now
        ABE=ABSER*ABSER
        REL=.5*RELER
      CALL GAUS2(BETA,YL,YU,REL,ABE,ROUND,ANSWR,IER,X,TGAS,PGAS,NBTYPE)
      END IF
C
      BETCAL=ANSWR
C
      IF (BETCAL.EQ.0.) WRITE (IPRNT,80) YL,YU,NBTYPE,INNER
80      FORMAT(' BETCAL)   YL=',1PG15.7,5X,'YU=',G15.7,5X,
     $            'NBTYPE=',I2,5X,'INNER=',I2)
C
C***             TRY TO CONTINUE EVEN IF INTEGRAL ESTIMATOR FAILS
C
      IF (IER.NE.0) THEN                     ! Trouble
C
        WRITE(IPRNT,4) INNER,NBTYPE,IER,X,YL,YU
    4 FORMAT(' INNER=',I3,'  INTEGRATION ERROR, NBTYPE =',I3,3X,'IER=',
     $  I3 /' OUTER VARIABLE=',1PE15.7,'  INNER DOMAIN=',2E15.7)
        DELVL=EXP(X)/BASESZ
        YMEAN=0.5*(YU+YL)
        ANSWR=(DELVL+0.5*DELVL*DELVL)*BETA(YMEAN,X,TGAS,PGAS,NBTYPE)
        ETEST2=ABS(YU-YL)/(ABS(YU)+ABS(YL))
        WRITE(IPRNT,14) ANSWR,ETEST,ETEST2,ROUND,DELVL
   14 FORMAT(' ANSWR=',1PE12.5,'  For ETEST=',2E12.3,' ROUND=',E12.5/
     $   ' Will Continue if DELVL of',E11.3,' < .01')
```

```
          IF (DELVL.GT.0.01) STOP
          BETCAL=ANSWR
        END IF
C
        RETURN
        END
C
C-------------------------------------------------------------------
C
        SUBROUTINE CALCON(QT,QVAP,SR,CONRAT,Z)
C
C*****************************************************************
C
C  PURPOSE:
C        To Calculate the Total Rate of Condensation (excluding Kelvin
C         Effect) and the Saturation Ratio as well as the current
C         Condensation Scaling Factor.
C
C  ON ENTRY:
C        QT(MMAX)           Total Mass in Each Size Section [kg/cu.m]
C        QVAP               Vapor Mass Concentration [kg/cu.m]
C        /AVGCOF/COEFAV     Array of Sectional Coefficients
C        /CONDNS/DELSAT     Reference SuperSaturation (for COEFAV) [-]
C        //      RATEG      Generation Rate of Condensible [kg/cu.m./sec]
C                            If DODVAP=.FALSE., RATEG is Condensation Rate also
C        /GAS/   TEMP       Temperature [K]
C        //      PSAT       Vapor Pressure [Pa]
C        /FLAGS/            Simulation Flags set here
C        /INDEX/ MS         Number of Size Sections
C        //      NGROW      Pointer to Growth Coefficients in COEFAV
C
C  ON RETURN:
C        SR                 Saturation Ratio of Condensible Species
C        CONRAT             Total Condensation Rate (no Kelvin) [kg/cu.m/sec]
C        Z                  Condensation Scaling Factor = (SR-1)/DELSAT
C
C  COMMENTS:
C        This routine must return SR, CONRAT, and Z under several different
C         possible constraints, such as fixed SR or fixed CONRAT.
C        The Kelvin Effect is (optionally) handled properly at
C         latter stages of the calculations, and supersedes Z calculation.
C        If SR<1., CONRAT=ZERO is returned.  (Doesn't evaluate evaporation.)
C
C*****************************************************************
C
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
      COMMON /AVGCOF/ COEFAV(NCMAX)      ! Sectional Coefficients
C       COMMON for Control Flags         AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
      COMMON /INDEX/  MS,KC,NQV,NQN,
     $  NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF    ! Pointers
      COMMON /CONDNS/ DELSAT     ! DELSAT for scaling
C
      DIMENSION QT(MMAX)         ! Total Mass per Size Section
      DATA NNEG / 0 /            ! Counter for Warnings (Negative Mass)
C
      Z=ZERO                     ! Initialize to No Condensation
      CONRAT=ZERO
      SR=ZERO
      IF (.NOT.DODVAP.AND..NOT.DOCOND) RETURN   ! No Condensation
C
C***            SUM FOR TOTAL MASS CONDENSING, WATCHING NEGATIVE TERMS
C
      COSUM=ZERO
      COBAD=ZERO
      DO I=1,MS                                 ! Sum over all sizes
        COTERM = COEFAV(NGROW+I) * QT(I)
        IF (QT(I).GT.ZERO) THEN
          COSUM=COSUM+COTERM             ! COSUM is Total Condensation Rate
        ELSE
          COBAD=COBAD+COTERM             ! Error due to negative QT
        END IF
      END DO
C
C***            CHECK FOR TROUBLE WITH EXCESSIVE NEGATIVE MASS TERMS
C
      IF (-COBAD.GE.COSUM.AND.COBAD.LT.ZERO) THEN
        NNEG=NNEG+1
        SR=SRATIO(QVAP)
        IF (NNEG.LE.20) WRITE(13,99) NNEG,SR     ! Extremely Unpromising
   99 FORMAT(/' DIRE WARNING - NEGATIVE COSUM IN CALCON  #',I5,
     $ '  with SR=',1PE10.2/)
        IF (NNEG.GE.500) STOP 'STOPPING ON 500 NEGATIVE COSUMS'
        RETURN  ! But it may be hopeless, but Return with no condensation
      ENDIF
C
C***            IS SATURATION RATIO KNOWN A PRIORI?
C
```

```
      IF (RATEG.LT.O.) THEN               ! Known PP of vapor
        SR=ONE+DELSAT                     ! DELSAT is current (and fixed)
        Z=ONE                             ! No scaling necessary
        CONRAT=COSUM
        RETURN
      END IF
C
C***            IS CONDENSATION RATE KNOWN A PRIORI?
C
      IF (.NOT.DODVAP) THEN               ! CONRAT is fixed
        IF (COSUM.LE.ZERO) THEN
          WRITE(13,98)
98    FORMAT(' DANGER -- COSUM IS ZERO WITH STEADY STATE CONDENSATION')
          WRITE(13,97) QT(1),COEFAV(NGROW+1)
97    FORMAT('    QT(1)=',1PE10.3,5X,'COEFAV(NGROW+1)=',E10.3)
          CONTINUE                        ! Condensation Rate is Zero
        ELSE
          CONRAT=RATEG                    ! CONRAT specified in this possibility
          Z=RATEG/COSUM                   ! Z scales condensation coefficients
          SR=ONE+Z*DELSAT
        END IF
        RETURN                            ! Steady State Vapor Concentration
      END IF
C
C***            USE VAPOR PHASE DIFFERENTIAL EQUATION (USUAL CASE)
C
      SR=SRATIO(QVAP)                     ! Calculate Saturation Ratio
      Z=(SR-ONE)/DELSAT                   ! Z scales con coef for true Delsat
      IF (NOEVAP.AND.Z.LE.ZERO) Z=ZERO    ! May supress Condensation
      CONRAT=Z*COSUM                      ! Net Condensation (without Kelvin eff

      IF (CONRAT.LT.ZERO) CONRAT=ZERO     ! Negative CONRAT is ambiguous
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE CALSIZ(DPMIN,DPMAX)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate Sectional Size Boundaries
C
C  ON ENTRY:
C       DPMIN                   Smallest Sectional Particle Diameter [m]
C       DPMAX                   Largest Sectional Particle Diameter [m]
C       /INDEX/ MS              Number of Size Sections
C
C  ON RETURN:
C       /SIZES/ DS(MMAX1)       Sectional Particle Diameter [m]
C       //      VS(MMAX1)       Sectional Particle Mass [kg]
C       /XSIZES/XS(MMAX1)       Sectional Log (Particle Mass)
```

```
C       //       DEL(MMAX)       Sectional Range in log(mass): XS(I-1)-XS(I)
C
C  COMMENTS:
C        Generates Geometically-Evenly Spaces Sections, so DEL is constant.
C        This is a convenient situation, but not necessary.
C
C***********************************************************************
C
      PARAMETER ( NEMAX = 218 )            ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                 ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )       ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )         ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )               ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )     ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )        ! MKS
      COMMON /INDEX/ MS                    ! Number of Size Sections
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)          ! Sectional Diam & Masses
      COMMON /XSIZES/ XS(MMAX1),DEL(MMAX)         ! Sectional Sizes II
C
      MS1=MS+1
      DS(1)=DPMIN                    ! Suggested: 30 Angstroms
      DS(MS1)=DPMAX                  ! Suggested:  3 Microns
C
      DO I=2,MS             ! Geometrically Equally Spaced Sections
        DS(I)=DS(1)*(DS(MS+1)/DS(1))**(FLOAT(I-1)/FLOAT(MS))
      END DO
C
      DO I=1,MS1
        VS(I)=ZERO          ! Tell RHODD to calculate Mass from Diameter
        CALL RHODD(VS(I),DS(I),RHO)
        XS(I)=ALOG(VS(I))           ! Calculate Logs of Sectional Particle Mass
      END DO
C
      DO L=1,MS             ! Calculate delta XS = log(particle mass) range
        DEL(L)=XS(L+1)-XS(L)        ! DEL = log(DPMAX/DPMIN) / MS
      END DO
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE CHECKE(TIME,DELTIM,Q,TGAS,PGAS,IPRNT,IFLAG,NEWCOF)
C
C***********************************************************************
C
C  PURPOSE:
C        To see that ESMAP variables have been set to reasonable values.
C        Program is stopped if input is unreasonable.
C
C  ON ENTRY:
```

```
C       All subroutine arguments must be set.
C       /TPSET/ must be set.
C       (See .DOC files for documentation on usage of variables.)
C
C  ON RETURN:
C       All variables unchanged.
C
C  COMMENTS:
C       None.
C
C*********************************************************************
 
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
      COMMON /INDEX/ MS,KC       ! Number of Size Sections and Components
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)        ! Sectional Diam & Masses
      COMMON /TPSET/ TGAS1,TGAS2,PGAS1,PGAS2    ! T,P set for interpolation
C       COMMON for Control Flags         AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      DIMENSION Q(NEMAX)
      DATA JONCE / 0 /
C
      ISTOP=0              ! Start with Flag O.K.
C
      IF (MS.LT.5.OR.MS.GT.MMAX) THEN
        ISTOP=1
        WRITE(IPRNT,2) MMAX
2     FORMAT(' --NUMBER OF SECTIONS MUST BE FROM 5 TO',I3)
      END IF
C
      IF (KC.LT.1.OR.KC.GT.8) THEN
        ISTOP=1
        WRITE(IPRNT,4)
4     FORMAT(' --NUMBER OF COMPONENTS MUST BE FROM 1 TO 8')
      END IF
C
      IF (DELTIM.LE.ZERO) THEN
```

```
          ISTOP=1
          WRITE(IPRNT,6)
 6      FORMAT(' --TIME STEP MUST BE POSITIVE')
        END IF
C
        IF (TGAS1.GE.TGAS2) THEN
          ISTOP=1
          WRITE(IPRNT,8)
 8      FORMAT(' --TEMPERATURE RANGE MUST BE POSITIVE')
        END IF
C
        IF (PGAS1.GE.PGAS2) THEN
          ISTOP=1
          WRITE(IPRNT,10)
 10     FORMAT(' --PRESSURE RANGE MUST BE POSITIVE')
        END IF
C
        IF (ROUND.GT.1.0) THEN
          ISTOP=1
          WRITE(IPRNT,12)
 12     FORMAT(' --ROUNDOFF ERROR MUST BE LESS THAN ONE')
        END IF
C
        IF ((IFLAG.LT.-1 .OR. IFLAG.GT.3) .AND. IFLAG.NE.7) THEN
          ISTOP=1
          WRITE(IPRNT,14) IFLAG
 14     FORMAT(' --IFLAG TO EP MAEROS MUST BE -1 thru 3, not',I3)
        END IF
C
        IF (IABS(NEWCOF).GT.15) THEN
          ISTOP=1
          WRITE(IPRNT,16)
 16     FORMAT(' --INVALID NEWCOF TO MAEROS')
        END IF
C
        DO I=1,MS
          IF (DS(I).LE.ZERO) THEN
            ISTOP=1
            WRITE(IPRNT,18) I
 18     FORMAT(' --PARTICLE DIAMETER AT LOWER BOUNDARY OF SECTION',I4,
      $        ' MUST BE POSITIVE')
          END IF
          IF (DS(I).GE.DS(I+1)) THEN
            ISTOP=1
            WRITE(IPRNT,20)
 20     FORMAT(' --PARTICLE DIAMETERS MUST BE IN ASCENDING ORDER')
          END IF
        END DO
C
        DO I=1,MS
          IF (VS(I+1).LT.2.*VS(I)) THEN
            IF (DOCOAG) ISTOP=1    ! Will allow if no coagulation.
            IF (JONCE.EQ.0) WRITE(IPRNT,22) I
```

```
22    FORMAT(' --PARTICLE DIAMETER NUMBER',I4
     $        ' DOES NOT SATISFY THE GEOMETRIC CONSTRAINT')
          JONCE=1
        END IF
      END DO
C
      X=ALOG(VS(1))
      Y=ALOG(VS(MS+1))
      F1=BETA(Y,X,TGAS,PGAS,4)
      F2=BETA(X,Y,TGAS,PGAS,4)
      IF (ABS(F1-F2)*1.E4.GT.ABS(F1)) THEN      ! Note Beta=0. is allowed
        ISTOP=1
        WRITE(IPRNT,24)
24    FORMAT(' --BETA ROUTINE IS NOT SYMMETRIC')
      END IF
C
      IF (F1.LT.ZERO.OR.F2.LT.ZERO) THEN
        ISTOP=1
        WRITE(IPRNT,26)
26    FORMAT(' --BETA ROUTINE IS NOT POSITIVE')
      END IF
C
      IF (ISTOP.NE.0) THEN
        WRITE(IPRNT,28)
28    FORMAT(' --CHECK TERMINATING RUN DUE TO INVALID INPUT TO MAEROS')
        STOP 'STOPPING DUE TO CHECK'
      END IF
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE COEF(NEWCOF,TGAS,PGAS,IPRNT)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Sectional Aerosol Coefficients
C
C  ON ENTRY:
C       NEWCOF          Flag Tells Which Coefficients are Needed:
C                               (See MAEROS for description)
C       TGAS            Gas Temperature [K]
C       PGAS            Gas Pressure [Pa]
C       IPRNT           Logical Unit Number for Output
C       /INDEX/ MS      Number of Size Sections
C       /SIZES/ VS      Particle Mass Array [kg]
C       /XSIZES/XS      Log of Particle Mass Array
C       //      DEL     Array containing XS range of section
C
C  ON RETURN:
C       /AVGCOF/ COEFAV() is set.
```

```
C
C  COMMENTS:
C       None.
C
C*************************************************************************
C
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                 ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )       ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )         ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )               ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /AVGCOF/ COEFAV(NCMAX)      ! Sectional Coefficients
      COMMON /INDEX/  MS,KC,NQV,NQN,
     $  NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF    ! Pointers
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)         ! Sectional Diam & Masses
      COMMON /XSIZES/ XS(MMAX1),DEL(MMAX)        ! Sectional Sizes II
C       COMMON for Control Flags        AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      COMMON /ROUND/ UROUND      ! Unit Round-Off Error (5.96E-8 for VAX REAL*4

C
      EXTERNAL BETCAL,DEPOST,GROWTH
C
      CALL SETGAS(TGAS,PGAS)             ! Set Gas Properties in /GAS/ COMMON
C
      REL=5.E-3
      ABSER=1.E-20
      MM1=MS-1
      MP1=MS+1
C
C      IF (NEWCOF.EQ.5.OR.NEWCOF.EQ.7.OR.NEWCOF.EQ.15) GO TO 5
C      IF (NEWCOF.EQ.6.OR.NEWCOF.EQ.9) GO TO 6
```

```
C
      IF (DOCOAG .AND. (NEWCOF.GE.1.AND.NEWCOF.LE.4 .OR.
     $    NEWCOF.GE.11.AND.NEWCOF.LE.14) ) THEN         ! Coagulation
C
C     NBTYPE = TYPE OF COEFFICIENT CALCULATED
C     INNER  = 0 INNER LIMITS OF INTEGRATION ARE CONSTANT
C              1 CHANGE LOWER INNER LIMIT OF INTEGRATION TO
C                 ALOG(BASESZ-OUTER INTEGRATION VARIABLE). IN THIS
C                 CASE FIXSZ IS THE INNER UPPER LIMIT OF INTEGRATION.
C              2 CHANGE UPPER INNER LIMIT OF INTEGRATION TO
C                 ALOG(BASESZ-OUTER INTEGRATION VARIABLE). IN THIS
C                 CASE FIXSZ IS THE INNER LOWER LIMIT OF INTEGRATION.
C
C
C       CALCULATE BETA(SUPER-1B,SUB-I,L-1,L)
C       STORE WITH I VARYING FIRST FROM 1 TO L-2
C
        NBTYPE=1
        INNER=1
        DO L=3,MS
          LM2=L-2
          LIBEF=(LM2*(L-3))/2
          DO I=1,LM2
            IER=1
            BASESZ=VS(L)
            FIXSZ=XS(L)
      CALL GAUSBT(BETCAL,XS(I),XS(I+1),REL,ABSER,UROUND,ANS,
     $            IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
            IF (IER.NE.0) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D           IF (ANS.EQ.0) WRITE(22,955) NBTYPE,I,L,LIBEF+I
 955  FORMAT(/' NBTYPE=',I2,5X,'I=',I3,5X,'L=',I3,5X,'COEFF #',I4)
            COEFAV(I+LIBEF)=ANS/(DEL(I)*(XS(L)-XS(L-1)))
          END DO
        END DO
C
C       CALCULATE BETA(SUPER-2A,SUB-I,L) AND BETA(SUPER-2B,SUB-I,L)
C       STORE WITH I VARYING FIRST FROM 1 TO L-1
C
        DO L=2,MS
          LM1=L-1
          LIBEF=(LM1*(L-2))/2
          DO I=1,LM1
            NBTYPE=2
            IER=1
            INNER=1
            BASESZ=VS(L+1)
            FIXSZ=XS(L+1)
      CALL GAUSBT(BETCAL,XS(I),XS(I+1),REL,ABSER,UROUND,ANS,
     $            IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
            IF (IER.NE.0) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D           IF (ANS.EQ.0) WRITE(22,955) NBTYPE,I,L,LIBEF+I+NB2A
            COEFAV(NB2A+I+LIBEF)=ANS/(DEL(I)*DEL(L))
            NBTYPE=3
            IER=1
            INNER=2
            BASESZ=VS(L+1)
            FIXSZ=XS(L)
      CALL GAUSBT(BETCAL,XS(I),XS(I+1),REL,ABSER,UROUND,ANS,
     $              IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
            IF (IER.NE.O) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D           IF (ANS.EQ.O) WRITE(22,955) NBTYPE,I,L,LIBEF+I+NB2B
            COEFAV(NB2B+I+LIBEF)=ANS/(DEL(I)*DEL(L))
          END DO
        END DO
C
C***              CALCULATE BETA(SUPER-3,SUB-L,L) IN THREE PARTS
C
        DO L=1,MS
          LP1=L+1
          NBTYPE=4
          IER=1
          INNER=1
          REL=1.E-2
          BASESZ=VS(LP1)
          FIXSZ=XS(LP1)
          ALV=ALOG(.5*VS(LP1))
      CALL GAUSBT(BETCAL,XS(L),ALV,REL,ABSER,UROUND,ANS,
     $              IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
          IF (IER.NE.O) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D         IF (ANS.EQ.O) WRITE(22,955) NBTYPE,I,L,NB3+L
C
          IER=1
          COEFAV(NB3+L)=ANS
          NBTYPE=4
          INNER=1
          ALV2=ALOG(VS(LP1)-VS(L))
          BASESZ=VS(LP1)
          FIXSZ=XS(LP1)
          CALL GAUSBT(BETCAL,ALV,ALV2,REL,ABSER,UROUND,ANS,
     $                  IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
          IF (IER.NE.O) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D         IF (ANS.EQ.O) WRITE(22,955) NBTYPE,I,L,NB3+L
          COEFAV(NB3+L)=ANS+COEFAV(NB3+L)
C
          IER=1
          NBTYPE=5
          INNER=O
          BASESZ=XS(L)
          FIXSZ=XS(LP1)
      CALL GAUSBT(BETCAL,ALV2,XS(LP1),REL,ABSER,UROUND,ANS,
     $              IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
          ANS=ANS+COEFAV(NB3+L)
          IF (IER.NE.O) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D         IF (ANS.EQ.O) WRITE(22,955) NBTYPE,I,L,NB3+L
          COEFAV(NB3+L)=.5*ANS/DEL(L)**2
        END DO
```

```
C
C     DETERMINE THE SECTIONAL COAGULATION COEFFICIENTS FOR
C     SCAVENGING OF PARTICLES IN SECTION L BY THOSE IN SECTION I
C     I.E. BETA(SUPER-4,SUB-I,L)
C     STORE WITH I VARYING FIRST FROM L+1 TO MS
C
          NBTYPE=6
          INNER=0
          DO L=1,MM1
            LP1=L+1
            NBEFR=((L-1)*(2*MS-L))/2
            DO I=LP1,MS
              INNER=0
              BASESZ=XS(L)
              FIXSZ=XS(LP1)
        CALL GAUSBT(BETCAL,XS(I),XS(I+1),REL,ABSER,UROUND,ANS,
     $             IER,IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
              IF (IER.NE.0) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D             IF (ANS.EQ.0) WRITE(22,955) NBTYPE,I,L,NB4+I-L+NBEFR
              COEFAV(NB4+I-L+NBEFR)=ANS/(DEL(I)*DEL(L))
            END DO
          END DO
        END IF                         ! Coagulation Done
C
C     DETERMINE THE SECTIONAL DEPOSITION COEFFICIENTS OF THE L-TH
C     SECTION ON THE J-TH DEPOSITION SURFACE
C
        IF (DODEPO.AND.NEWCOF.NE.6.AND.NEWCOF.NE.9) THEN
    5     REL=1.E-3
          DO L=1,MS
            DO J=1,3
              NBTYPE=J+6
              IER=1
              CALL GAUS2(DEPOST,XS(L),XS(L+1),REL,ABSER,UROUND,ANS,
     $                   IER,DUM,TGAS,PGAS,NBTYPE)
              INDEXW=NDEPST+J+3*(L-1)
              IF (IER.NE.0) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
              COEFAV(INDEXW)=ANS/DEL(L)
C  This can go to zero sometimes, e.g., large particles onto ceiling.
D             IF (ANS.EQ.0) WRITE(22,955) NBTYPE,I,L,INDEXW
D             TYPE 986, ANS,INDEXW,DEL(L)
D 986    FORMAT(' ANS=',1PE10.2,4X,'INDEXW=',I3,4X,'DEL(L)=',1PE10.2)
            END DO
          END DO
        END IF
C
        IF (NEWCOF.NE.6.AND.NEWCOF.LT.10) THEN     ! Condensation
C
C     CALCULATE THE SECTIONAL CONDENSATION COEFFICIENTS,
C     G(SUPER-1,SUB-L,KC),G(SUPER-2A,SUB-L,KC) AND
C     G(SUPER-2B,SUB-L,KC) FOR L=1,...,MS
C
    6     DO L=1,MS
```

```
          NBTYPE=10
          IER=1
          CALL GAUS2(GROWTH,XS(L),XS(L+1),REL,ABSER,UROUND,ANS,
     $                   IER,DUM,TGAS,PGAS,NBTYPE)
          IF (IER.NE.0) CALL ERRORO(IER,NBTYPE,ANS,IPRNT)
D         IF (ANS.EQ.0) WRITE(22,955) NBTYPE,I,L,NGROW+L
          COEFAV(NGROW+L)=ANS/DEL(L)
        END DO
C
        DO L=1,MM1
          DUM = GROWTH(XS(L+1),DUMMY,TGAS,PGAS,NBTYPE) /
     $           ( (DEL(L+1)+DEL(L)) * DEL(L+1) * DEL(L) )
          COEFAV(NGROW+MS+2*L-1)=DUM*DEL(L+1)**2
          COEFAV(NGROW+MS+2*L)=DUM*DEL(L)**2
        END DO
        COEFAV(NGROW+3*MS-1) = GROWTH(XS(MS+1),DUM,TGAS,PGAS,NBTYPE)
     $                            / ( 2.*DEL(MS) )
      END IF
C
      RETURN
      END


      SUBROUTINE ERRORO(IER,NBTYPE,ANS,IPRNT)    ! Reports Integration Error
      WRITE(IPRNT,5) IER,NBTYPE,ANS
5     FORMAT(//' OUTER INTEGRATION ERROR NUMBER',I3,2X,
     $          'FOR COEFFICIENT TYPE',I3,2X,'RETURNED',1PE13.7)
      RETURN            ! Or STOP
      END
C
C------------------------------------------------------------------
C
      FUNCTION DEPOST(X,DUMMY,TGAS,PGAS,NBTYPE)
C
C******************************************************************
C
C  PURPOSE:
C       To Calculate the Depostion Coefficients Due To
C        Gravity and Diffusion.  The Coefficient is Given
C        For the Overall Container.
C
C  ON ENTRY:
C       X                Log of Particle Mass [ln(kg)]
C       DUMMY            Not Used
C       TGAS             Gas Temperature [K]
C       PGAS             Gas Total Pressure [Pa]
C       NBTYPE           Type of Sectional Coefficient; Should Be 7 Here
C       /GAS/   DENAIR   Background Gas Density [kg/cu.m]
C       //      FREEMP   Background Gas Mean Free Path [m]
C       //      VISCOS   Background Gas Viscosity [kg/m/sec]
C       //      DELDEP   Boundary Layer [m] or Flag (if negative):
C                            -1  Spherical Container
C                            -2  Box with equal length sides
```

```
C                         <= -4  No Deposition
C
C  ON RETURN:
C       DEPOST            Deposition Coefficient
C
C  COMMENTS:
C       None.
C
C***********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )        ! MKS
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
      IF (DELDEP.GT.0.) THEN              ! Boundary Layer Theory
        DEPOST=OLDDEP(X,DUMMY,TGAS,PGAS,NBTYPE) ! From Old MAEROS Package
        RETURN
      END IF
C
      DEPOST=ZERO
      IF (DELDEP.LE.-4.) RETURN           ! Code for No Deposition
      IF (NBTYPE.EQ.8 .OR. NBTYPE.EQ.9) RETURN  ! Unused Array Space
      IF (NBTYPE.NE.7) STOP 'ILLEGAL NBTYPE TO NEW DEPOST'
C
      V=EXP(X)                    ! Mass
      D=0.
      CALL RHODD(V,D,RHO)         ! Calculate Particle Diameter
C
C***             AIR VISCOSITY, DENSITY, MEAN FREE PATH HELD IN /GAS/
C***              DOUBLECHECK TEMPERATURE & PRESSURE ARE CONSISTENT
C
      IF (TGAS.NE.TEMP.OR.PGAS.NE.PRES) THEN
        IF (TGAS.NE.TEMP) TYPE 21, TEMP,TGAS
21      FORMAT(/' WARNING: /GAS/ TEMP =',F7.1,' while TGAS=',F7.1 /)
        IF (PGAS.NE.PRES) TYPE 22, PRES,PGAS
22      FORMAT(/' WARNING: /GAS/ PRES =',1PE9.2,' while PGAS=',E9.2 /)
        CALL SETGAS(TGAS,PGAS)
      END IF
C
      FCHI=CHI
      AKN=2.*FREEMP/D             ! Knudsen Number of particle in air
      BMOBIL=1.+AKN*(FSLIP+.4*EXP(-1.1/AKN))
      VTERM=.544*RHO*D*D*BMOBIL/VISCOS  ! Terminal Velocity
      DIF=1.46E-24*TGAS*BMOBIL/(VISCOS*FCHI*D)
      SQRKED=SQRT(DIF*AKE)
      IF (DELDEP.EQ.-1.) THEN
        Z=0.5*PI*VTERM/SQRKED                ! This is a Sphere
        DIAMET=(6.*VOLUME/PI)**(1./3.)  ! Container Diameter
        DEPOST=6.*SQRKED*(2.*DEBYE1(Z)+0.5*Z)/(PI*DIAMET)
D       REF=3.*VTERM/(2.*DIAMET)                  ! If Settling Only
D       TYPE 77, 1.E6*D,1.E2*VTERM,DIAMET,DEPOST,REF
D 77    FORMAT(' ',F10.3,' uM',5X,1PE10.2,'cm/sec',5X,1P3E10.2)
      ELSE IF (DELDEP.EQ.-2.) THEN ! Corner & Pendlebury: Formula for a Cube
        DIAMET=VOLUME**(1./3.)           ! Length of Side of Cube
        DEPOST=(8.*SQRKED/PI+VTERM*COTH(PI*VTERM/4./SQRKED))/DIAMET
      ELSE
        STOP 'BAD VALUE OF DELDEP'
      END IF
C
      RETURN
      END


      FUNCTION COTH(X)
      COTH=(EXP(X)-EXP(-X))/(EXP(X)+EXP(-X))
      RETURN
      END


      FUNCTION DEBYE1(X)
      DIMENSION X1(43),D1(43)
C     Compute Debye Function, Order 1, By Interpolating a Look-up Table
C     The linear interpolation should be accurate to nearly 4 sig figs
      DATA NTABLE /43/  ! Dimensioning for # of points in table
      DATA X1 / 0.0 , 0.1 , 0.2 , 0.3 , 0.4 , 0.5 , 0.6 , 0.7 ,
     $ 0.8 , 0.9 , 1.0 , 1.1 , 1.2 , 1.3 , 1.4 , 1.6 , 1.8 ,
     $ 2.0 , 2.2 , 2.4 , 2.6 , 2.8 , 3.0 , 3.2 , 3.4 , 3.6 ,
     $ 3.8 , 4.0 , 4.2 , 4.4 , 4.6 , 4.8 , 5.0 , 5.5 , 6.0 ,
     $ 6.5 , 7.0 , 7.5 , 8.0 , 8.5 , 9.0 , 9.5 , 10.0 /
      DATA D1 / 1.00,0.9753,0.9511,0.9275,0.9044,0.8819,0.8600,0.8385,
     $ 0.8177,0.7973,0.7775,0.7582,0.7394,0.7212,0.7034,0.6694,0.6372,
     $ 0.6069,0.5784,0.5516,0.5264,0.5027,0.4804,0.4596,0.4400,0.4216,
     $ 0.4043,0.3881,0.3730,0.3587,0.3453,0.3327,0.3209,0.2942,0.2713,
     $ 0.2513,0.2339,0.2187,0.2052,0.1933,0.1826,0.1731,0.1644 /
      IF (X.LT.X1(1)) THEN
        TYPE 55
55      FORMAT(/' BAD ARGUMENT TO DEBYE1'/)
        DEBYE1=1.
        RETURN
      END IF
      I=2
      DO WHILE (X.GE.X1(I))
        I=I+1
        IF (I.GT.NTABLE) THEN
          DEBYE1=D1(NTABLE)*X1(NTABLE)/X
          RETURN
        END IF
      END DO
      DEBYE1=D1(I)+(X-X1(I))*(D1(I)-D1(I-1))/(X1(I)-X1(I-1))
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE DIFFUN(NEQ,T,Q,DQDT)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Derivatives dQ/dt for the Multicomponent
C        Aerosol Model.
C
C  ON ENTRY:
C       NEQ     Number of elements in Q or DQDT (augmented) arrays
C       T       Time at which derivatives are to be evaluated [sec]
C       Q       Array of Sectional Masses [kg/cu.m.]
C
C  ON RETURN:
C       DQDT    Array of Sectional Mass Time Derivatives [kg/cu.m/sec]
C
C  COMMENTS:
C
C     THIS ROUTINE CALCULATES THE DERIVATIVES (I.E. EQUATION 50 OF
C     THE PAPER 'SIMULATION OF MULTICOMPONENT AEROSOL DYNAMICS',
C     FRED GELBARD AND JOHN H. SEINFELD, J. COLLOID AND INTERFACE
C     SCIENCE, VOL.78,P.485,1980)
C
C*********************************************************************
C*  ORIGINAL ISSUED BY SANDIA LABORATORIES, WRITTEN BY FRED GELBARD  *
C*********************************************************************
C---------------------------------------------------------------------
C   Modifications by Dale R. Warren for use on Caltech VAX 11/780
C   This is June 1984 Version -- Number Conservation with Standard COEFAV
C   Uses Nucleation Package J for homogeneous nucleation
C   Options controlled by /FLAGS/ (set in file CHOOSE.INC)
C   Modified to allow Evaporation . . . SSK < 0 for Possible Evaporation
C   Uses SSKELV to compute the Kelvin Effect (effective supersaturation)
C---------------------------------------------------------------------
C
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)         ! Sectional Diam & Masses
C       COMMON for Control Flags         AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
      COMMON /INDEX/  MS,KC,NQV,NQN,
     $  NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF    ! Pointers
      COMMON /AVGCOF/ COEFAV(NCMAX)      ! Sectional Coefficients
      COMMON /PSRATE/  PSRATE(NEMAX)     ! Sectional Particle Source Rates
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
      COMMON /NPASS/ TIME        ! For Optional Nucleation Output
      COMMON /DF2/ CONKEL,RJM    ! kg/cu.m/sec
C
      LOGICAL DEBUG /.FALSE./
      REAL*4 Q(NEMAX),DQDT(NEMAX),QT(MMAX),QTOT
      REAL*4 GAIN(MMAX)
      REAL*4 QVAP,DQVAP
      REAL*4 Z,CONRAT,SR
      REAL*4 CONKEL,SCON,DIKELV,SUM
      REAL*4 TCON,FM,FP,FM1,FM2,FP1,FP2
      REAL*4 RALOSS,TRANS(10)    ! Assume no more than 10 components
      REAL*4 COEF1,COEF2
      DATA RJMMIN / 1.E-30 /     ! Minimum mass rate of nucleation
      DATA QMIN / 0. /           ! Minimum significant Q mass
      DATA GEOMET / TWO /        ! Needed if GEOSEC is .TRUE.
C
C
      DIKELV=4.*SURTEN*CONMW/(DENSTY*RGAS*TEMP)          ! Kelvin diameter
C
C***             SUM TOTAL MASS CONCENTRATION OF SECTION L AND PUT IN QT(L)
C
      QTOT=ZERO
      DO L=1,MS            ! Section L from 1 thru MS
        SUM=ZERO
        LQ=(L-1)*KC        ! Subscript Base of Section L in Q's
        DO K=1,KC          ! Component K from 1 thru KC
          I=K+LQ           ! Index (K,L)
          DQDT(I)=ZERO     ! Necessary Initialization
          SUM=SUM+Q(I)     ! Sum Mass in Section L
        END DO
        QT(L)=SUM          ! Total Mass Concentration in Section L
```

```
          QTOT=QTOT+SUM    ! Sum All Particle Mass
        END DO
C
        IF (DOCOND) THEN
          DO K=1,KC        ! For Each Component
            TRANS(K)=ZERO ! Initialize Intersectional Flux to 0
          END DO
        END IF
C
C***              COMPUTE EFFECT OF REMOVAL MECHANISMS
C
        IF (DODEPO.AND.QTOT.GT.QMIN) THEN ! Include Deposition
          DO L=1,MS
            LDEP=3*(L-1)+NDEPST
            TOTDEP=0.
            DO J=1,3
              TOTDEP=TOTDEP+COEFAV(LDEP+J)
            END DO
            DO K=1,KC
              I=K+(L-1)*KC                   ! Index (K,L)
              DQDT(I)=DQDT(I)-TOTDEP*Q(I)
            END DO
          END DO
        END IF
C
C***              INCLUDE EXPLICIT PARTICLE SOURCES
C
        IF (DOSORC) THEN             ! Include Particle Mass Sources
          DO L=1,MS
            DO K=1,KC
              I=K+(L-1)*KC                   ! Index (K,L)
              DQDT(I)=DQDT(I)+PSRATE(I)      ! PSRATE must be set elsewhere
            END DO
          END DO
        END IF
C
C***              CALCULATE THE CHANGE DUE TO COAGULATION
C
C-------------------- Start of Coagulation Code Block --------------------
C
        IF (DOCOAG.AND.QTOT.GT.QMIN) THEN
         DO 30 L=1,MS      ! For Section L from 1 thru MS
          LM1=L-1
          LM2=L-2
          LQ=LM1*KC        ! Subscript Base of Section L in Q's
          LMQ=LM2*KC       ! Subscript Base of Section L-1 in Q's
          LC=(LM1*LM2)/2   ! Subscript Base of Section L in COEFF (type 1,2)
          LMC=((L-3)*LM2)/2        ! Subscript Base of Section L-1 in COEFF
C
          DO 30 K=1,KC     ! For Component K from 1 thru KC
           IM=K+LMQ                ! Index (K,L-1)
           I=K+LQ          ! Index (K,L)
           IP=K+LPQ                ! Index (K,L+1)
          SUM=ZERO
          IF (L.GE.3) THEN          ! ( small + L-1 ==> L }
           DO J=1,LM2     ! Section J for small sections up to L-2
            IJ=(J-1)*KC+K           ! Index (K,J)
            SUM=SUM+QT(J)*(COEFAV(NB2A+J+LMC)*Q(IM)
     $      -COEFAV(NB2A+J+LC)*Q(I))
     $      +Q(IJ)*(COEFAV(J+LMC)*QT(LM1)
     $      +COEFAV(NB2B+J+LC)*QT(L))
           END DO
          END IF
          IF (L.GT.1) SUM = SUM+QT(LM1)*(COEFAV(NB3+LM1)*Q(IM)
     $                - COEFAV(NB2A+LM1+LC)*Q(I))
     $                + COEFAV(NB2B+LM1+LC)*QT(L)*Q(IM)
   30     DQDT(I) = SUM - COEFAV(NB3+L)*QT(L)*Q(I)
C
C***             CALCULATE REMOVAL RATE FROM A SECTION DUE TO SCAVENGING
C***               BY HIGHER SECTIONS   (COAGULATION)
C
        MS1=MS-1
        DO 40 L=1,MS1     ! Section L from 1 thru MS-1
         LM1=L-1
         LQ=LM1*KC                 ! Subscript Base of Section L in Q's
         LBF=(LM1*(2*MS-L))/2
         SUM=ZERO
         LP1=L+1
         DO 35 J=LP1,MS  ! Consider sections J from L+1 thru MS
   35     SUM=SUM+COEFAV(NB4+LBF+J-L)*QT(J)
         DO 40 K=1,KC     ! Component K from 1 thru KC
          I=K+LQ          ! Index (K,L)
   40     DQDT(I)=DQDT(I)-SUM*Q(I)
      ENDIF
C
C-------------------- End of Coagulation Code Block --------------------
C
C***             CALCULATE THE CONDENSATIONAL GROWTH FACTORS
C
   50 QVAP=Q(NQV)        ! The Q array holds true QVAP, needed by CALCON
      CALL CALCON(QT,QVAP,SR,CONRAT,Z)           ! Calculate SR,CONRAT,Z
C       Z is a scaling factor for the Condensational Growth Rate Coefficients
C       Z = actual pressure driving force / reference pressure difference
C       This program assumes Z will be positive.  For Z=0 or SR=1,
C        the Kelvin effect can not readily be included.
C
C***             CALCULATE THE EFFECTS OF INTRA-SECTIONAL CONDENSATIONAL GROWTH
C
C       Note that this only occurs for the condensing component
C       Note also that this is the only growth term that changes
C        the overall mass present (except finite domain error
C        and error introduced in stabilizing against negative mass)
C
      CONKEL=ZERO
      IF (QTOT.LT.QMIN) WRITE(66,*) ' QTOT < 0.'
      IF (DOCOND.AND.QTOT.GT.QMIN) THEN
```

```
C
      DO 65 L=1,MS             ! Section L from 1 thru MS
       I=L*KC                  ! Index (KC,L) -- Last Component Only
       IF (QT(L).GT.ZERO) THEN
        SCONO=COEFAV(NGROW+L)*QT(L)    ! Reference Sectional Condensation
        IF (DOKELV) THEN
         DMEAN=SQRT(DS(L)*DS(L+1))     ! Use Geometric Mean Section Diameter
         SSK=SSKELV(SR,DMEAN,DIKELV)
         IF (SSK.LE.ZERO .AND. Q(I).LE.QMIN) THEN
          SCON=ZERO            ! No Condensation and Nothing to Evaporate
         ELSE IF (SSK.GT.ZERO) THEN
          SCON=(SSK/DELSAT)*SCONO
         ELSE
          SCON=(SSK/DELSAT)*COEFAV(NGROW+L)*QT(L)       ! Evaporation
C  Evaporation treated as if each particle had volatile shell;
C  Use Q(I) rather that QT(L) if separate particles for each component.
          IF (NOEVAP) SCON=ZERO
         ENDIF
         CONKEL=CONKEL+SCON            ! Keep track of total condensation
        ELSE                           ! If Kelvin Effect Neglected
         SCON = Z * SCONO              ! Z == ( SR - ONE ) / DELSAT
        ENDIF
        DQDT(I)=DQDT(I)+SCON
        GAIN(L)=SCON                   ! Save Intrasectional Terms
       ELSE
        GAIN(L)=ZERO                   ! If no positive mass
       ENDIF
   65  CONTINUE
C
      ENDIF
C
      IF (DOCOND.AND.DODVAP) THEN
       IF (DOKELV) THEN
        DQVAP=RATEG-CONKEL       ! Rate of Change of QVAP with time
D        EFFECT=ZERO
D        DCRIT=DIKELV/ALOG(SR)
D       IF CONRAT.NE.ZERO) EFFECT=CONKEL/CONRAT
D        WRITE(12,112) EFFECT,DCRIT,T
D  112 FORMAT(' Kelvin Effect =',F10.5,'  D*=',1PE11.3,' M   after',
D     $       0PF10.2,' Seconds')
       ELSE
        DQVAP=RATEG-CONRAT       ! Neglects Kelvin Effect
       ENDIF
      ENDIF
C
C
C***            CALCULATE THE EFFECTS OF HOMOGENEOUS NUCLEATION
C
      IF (DONUCL.AND.SR.GT.ONE) THEN
       TIME=T                           ! Pass time to NUCL for debugging
       RJM=RJMMIN                       ! Pass a rate considered negligible
       CALL JMKS(SR,RJM,CONRAT,GCRIT)   ! Nucleation Mass Rate in MKS units
D      WRITE(11,66) RJM,GCRIT,SR
```

```
    66  FORMAT('  RJM=',1PE12.3,5X,'Gcrit=',E12.3,5X,
      $ 'SR=',0PF15.7)
        IF (RJM.LE.RJMMIN) GOTO 70
        DQDT(KC)=DQDT(KC)+RJM    ! Last Component, First Section
        DQVAP=DQVAP-RJM
        TRANS(KC)=RJM
        IF (MS*KC+2.EQ.NEQ)  DQDT(NEQ)=RJM  ! If following total nucleation
C
D       TYPE 68, DQDT(KC),DQVAP,Q(KC),KC
D    68 FORMAT(' DQDT(KC)=',1PE11.3,4X,'DQVAP=',E11.3,4X,
D     $ 'Q(KC)=',E11.3,4X,'KC=',I3)
C
      ENDIF
C
C***             CALCULATE INTER-SECTIONAL CONDENSATIONAL GROWTH
C
C        Negative Mass is Treated as Zero Mass in Q or QT arrays
C
   70 NPM=NGROW+MS
C
C---------- Start of Intersectional Condensation Code Block ----------
C
      IF (DOCOND .AND. QTOT.GT.QMIN) THEN
C
       L=1                               ! Handle Smallest Section Here
       IF (DOKELV) THEN                  ! Calculate Kelvin Effect
         IF (DOCON2) THEN
           DMEAN=DS(2)
         ELSE
           DMEAN=SQRT(DS(1)*DS(2))
         ENDIF
         SSK=SSKELV(SR,DMEAN,DIKELV)
       ELSE
          SSK=SR-ONE                     ! Neglect Kelvin Effect
       ENDIF
C
       IF (DONCON) THEN                  ! New Number Conserving
         DELX=ALOG(VS(2)/VS(1))          ! May use if geometrically even
         IF (SSK.GE.ZERO) THEN           ! Condensation
           DELXS=ALOG(VS(3)/VS(1))/TWO   ! For Generalized Spacing
           FACTOR = ONE - EXP(-DELXS)
         ELSE                    ! Evaporation
           FACTOR = ONE - EXP(DELX)
         ENDIF
         COEF1=COEFAV(NGROW+1)/FACTOR/TWO
         COEF2=COEF1     ! Irrelevant -- DONCON works only with 1st Order
       ELSE                              ! Old Original MAEROS Form
         COEF1=COEFAV(NPM+1)             ! Standard 1st or 2nd Order
         COEF2=COEFAV(NPM+2)             ! 2nd Order
       ENDIF
       IF (SSK.LT.ZERO.AND.(NOEVAP.OR.Q(KC).LE.QMIN)) THEN
         COEF1=ZERO
         COEF2=ZERO
        ENDIF
 C
        DO 80 I=1,KC        ! Index (K,1) -- Smallest Section Only
         IF (.NOT.GEOSEC)
     $      GEOMET=ALOG(DS(3)/DS(1))/ALOG(DS(2)/DS(1))
         IP=I+KC            ! Index (K,2)
         FP1=AMAX1(COEF1*Q(I),ZERO)*SSK/DELSAT    ! Negative Mass won't grow
         FP2=AMAX1(COEF2*Q(IP),ZERO)*SSK/DELSAT   ! Used in 2nd Order Model
         IF (DOCON2) THEN
 C
 C----------------------2nd ORDER--------------------------------------
          RALOSS=(FP1+FP2)-DQDT(I)                 ! 2nd Order -DQDT(I) total
          IF (Q(I).LE.ZERO) THEN
              FP=ZERO                              ! Nothing can leave
          ELSE IF (DOLIMT.AND.Q(I)/TCON.LT.RALOSS) THEN
              FP=AMIN1((DQDT(I)+Q(I)/TCON),FP1+FP2)
 C Designed to prevent excessive stiffness and avoid negative mass sections
 C Without this modification the second order was untenable computationally
              IF (FP.LT.FP1) THEN
                WRITE (13,89) T,I,FP,FP1
                FP=AMIN1(GEOMET*FP1,FP1+FP2)
              ENDIF
    89 FORMAT(' FP TROUBLE, T=',F9.2,3X,'I=',I3,1P2E14.3)
          ELSE
            FP=FP1+FP2   ! Linear Interpolation (pure 2nd Order)
          ENDIF
 C--------------------------------------------------------------------
        ELSE                ! 1st Order, DOCON2=.FALSE.
          FP=FP1*GEOMET ! GEOMET=2. if geometrically evenly spaced sections
        ENDIF
        IF (SSK.LT.ZERO.AND.I.NE.KC) FP=ZERO     ! Solid Nuclei Remain Here
        IF (FP.LT.ZERO) TYPE *, ' Warning: Negative Intersectional Mass'
        IF (DONCON.AND.Q(I).GT.ZERO.AND.QT(1).GT.ZERO) THEN
          FPX = GAIN(1) / FACTOR * (Q(I)/QT(1))
c         WRITE(66,665) I,1,GAIN(1),FACTOR,Q(I)/QT(1),FP,FPX
c         WRITE(66,666) I,1,SSK,FP,FPX,Q(I)
        END IF
        DQDT(I)=DQDT(I)-FP
        IF (SSK.LT.ZERO) THEN    ! Evaporation Case
          IF (I.EQ.KC) DQVAP=DQVAP+FP    ! Subsectional particles go to vapor
          TRANS(I)=ZERO
        ELSE                      ! Normal Condensation Case
          TRANS(I)=FP             ! Intersectional mass out of sect 1 by comp I
        ENDIF
   80  CONTINUE
C
C
      DO 200 L=2,MS        ! For Section L from 2 thru M
C
       IF (.NOT.GEOSEC) THEN
         IF (L.NE.MS) THEN
          GEOMET=ALOG(DS(L+2)/DS(L))/ALOG(DS(L+1)/DS(L))
         ELSE
```

```
            GEOMET=TWO
          ENDIF
          IF (ABS(GEOMET-TWO).GE.5.E-6) TYPE *,'BAD GEOMET = ',GEOMET
        ENDIF
C
        NPM=NGROW+MS+2*L-2
        LPQ=L*KC                  ! Q subscript for Section L+1
        LQ=LPQ-KC                 ! Q subscript for Section L
        LMQ=LQ-KC                 ! Q subscript for Section L-1
C
        IF (DOKELV) THEN          ! Calculate Kelvin Effect
          IF (DONCON) THEN
           DMEAN=SQRT(DS(L)*DS(L+1))
          ELSE IF (DOCON2) THEN
           DMEAN=DS(L+1)
          ELSE
           DMEAN=SQRT(DS(L)*DS(L+1))
          ENDIF
          SSK=SSKELV(SR,DMEAN,DIKELV)
        ELSE
          SSK=SR-ONE              ! If no Kelvin Effect Included
        ENDIF
C
C        IF (SSK.LT.ZERO.AND.(NOEVAP.OR.Q(LQ+KC).LE.QMIN)) THEN
C
        COEF1=COEFAV(NPM+1)
        COEF2=COEFAV(NPM+2)
        IF (DONCON) THEN          ! Intrasectional fixes intersectional
          DELX=ALOG(VS(L+1)/VS(L))                          ! Fine if GEOSEC
          IF (SSK.GE.ZERO) THEN
           IF (L.NE.MS) DELXS=ALOG(VS(L+2)/VS(L))/TWO     ! Generalized
           IF (L.EQ.MS) DELXS=DELX        ! Otherwise can't handle last section
           FACTOR = ONE - EXP(-DELXS)
          ELSE                    ! Evaporation
           DELXS=ALOG(VS(L+1)/VS(L-1))/TWO
           FACTOR = ONE - EXP(DELXS)             ! Negative for Evaporation
          ENDIF
          COEF1=COEFAV(NGROW+L)/FACTOR/TWO
          COEF2=COEF1             ! No consistent way to handle this; use Order
          IF (DEBUG) THEN
           WRITE(25,777) L,COEFAV(NGROW+L),2.*COEFAV(NPM+1),2.*COEF1
  777      FORMAT(' L=',I3,5X,'COEF=',1P3E15.5)
          ENDIF
        ENDIF
        IF (SSK.LT.ZERO.AND.(NOEVAP.OR.Q(LPQ).LE.QMIN)) THEN
          COEF1=ZERO
          COEF2=ZERO
        ENDIF
C
        DO 100 K=1,KC      ! For component K from 1 thru KC
         I=K+LQ                     ! Index (K,L)
         IP=K+LPQ                   ! Index (K,L+1)
         DQDT(I)=DQDT(I)+TRANS(K)          ! Add flux from lower section growth
         FP1=COEF1*AMAX1(Q(I),ZERO)*SSK/DELSAT
         IF (DOCON2) THEN
          FP2=COEF2*AMAX1(Q(IP),ZERO)*SSK/DELSAT
          RALOSS=(FP1+FP2)-DQDT(I)         ! Order 2 Total -DQDT(I)
          IF (Q(I).LE.ZERO) THEN
             FP=ZERO
          ELSE IF (DOLIMT.AND.Q(I)/TCON.LT.RALOSS) THEN
              FP=AMIN1((DQDT(I)+Q(I)/TCON),FP1+FP2)
              IF (FP.LT.FP1) THEN
                WRITE (13,89) T,I,FP,FP1
                FP=AMIN1(GEOMET*FP1,FP1+FP2)
              ENDIF
          ELSE
             FP=FP1+FP2
          ENDIF
        ELSE                        ! Recommended First Order Condensation
          FP=FP1*GEOMET
        ENDIF
C
        IF (FP.LT.ZERO) TYPE *, ' Warning: FP<0.'
        IF (DONCON.AND.Q(I).NE.ZERO.AND.QT(L).NE.ZERO) THEN
          FPX = GAIN(L) / FACTOR * (Q(I)/QT(L))
c         WRITE(66,665) I,L,GAIN(L),FACTOR,Q(I)/QT(L),FP,FPX
665       FORMAT(1X,2I4,' G=',1PE9.2,' F=',E9.2,' R=',E9.2,
        1   ' FP=',E9.2,' FPX=',E9.2)
c         WRITE(66,666) I,L,SSK,FP,FPX,Q(I),GAIN(L),Q(I)/QT(L)
666       FORMAT(' I=',I3,3X,'L=',I3,3X,'SSK=',1PE9.2,' FP=',E9.2,
        1   ' FPX=',E9.2,' Q(I)=',E9.2,' GAIN=',E9.2,' QR=',E9.2)
        END IF
        IF (SSK.GE.ZERO) THEN
          TRANS(K)=FP               ! Calculate flux to higher section
        ELSE
          TRANS(K)=ZERO
          IM=K+LMQ                  ! Index (K,L-1)
          DQDT(IM)=DQDT(IM)+FP      ! Evaporation Adds to previous section
        ENDIF
  100   DQDT(I)=DQDT(I)-FP
C
```

```
  200  CONTINUE
C
        IF (DEBUG.AND.SR.GT.ONE) THEN
          DEBUG=.FALSE.
          CLOSE (25,STATUS='SAVE')
        ENDIF
C
      ENDIF
C
C-------------------- End of Condensation Code Block --------------------
C
C
      IF (DODVAP) DQDT(NQV)=DQVAP        ! Extra Section is Vapor Phase
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE GAUS2(F,XL,XU,RELER,ABSER,ROUND,ANSWR,IER,EXTRA1,
     $EXTRA2,EXTRA3,NEXTRA)
C
C     THIS ROUTINE COMPUTES THE INTEGRAL OF F(X,EXTRA1,EXTRA2,EXTRA3,
C     NEXTRA) FROM XL TO XU.  A TWO POINT GAUSS-LEGENDRE QUADRATURE
C     FORMULA IS USED. CONVERGENCE IS CHECKED BY DIVIDING THE DOMAIN IN
C     HALF AND REAPPLYING THE FORMULA IN EACH HALF.  IF THE VALUE OF THE
C     INTEGRAL CALCULATED OVER THE ENTIRE DOMAIN IS NOT EQUAL TO THE
C     SUM OF THE INTEGRALS IN EACH HALF (WITHIN THE
C     USER SPECIFIED ERROR TOLERANCE), EACH HALF IS FURTHER DIVIDED
C     INTO HALVES AND THE GAUSS-LEGENDRE FORMULA IS REAPPLIED. THE
C     PROCEDURE WILL CONTINUE ITERATING (I.E. SUBDIVIDING),UNTIL
C     CONVERGENCE IS ACHIEVED OR THE MAXIMUM NUMBER OF ITERATIONS IS
C     REACHED.  THE MAXIMUM NUMBER OF ITERATIONS IS EITHER THE SET
C     DEFAULT VALUE OF 20 (WHERE THE FIRST ITERATION IS FOR EVALUATION
C     OVER THE ENTIRE DOMAIN), OR THE LARGEST NUMBER OF ITERATIONS
C     POSSIBLE WITHOUT SEVERE MACHINE ROUND-OFF ERRORS, WHICHEVER IS
C     SMALLER.  THE MACHINE ROUND-OFF ERROR CHECK IS MADE TO INSURE
C     THAT THE INTEGRATION DOMAIN IS NOT TOO SMALL SO AS TO BE
C     INSIGNIFICANT.  SINCE THE PROCEDURE IS ADAPTIVE, ONLY THE REGIONS
C     WHICH ARE NONCONVERGENT ARE DIVIDED INTO HALVES. THIS CODE WAS
C     WAS WRITTEN BY FRED GELBARD, FEBRUARY, 1982.
C
C                          CALLING SEQUENCE
C
C     CALL GAUS2(F,XL,XU,RELER,ABSER,ROUND,ANSWR,IER,EXTRA1,EXTRA2,
C                EXTRA3,NEXTRA)
C
C     NOTE:THE USER MUST SUPPLY A FUNCTION SUBROUTINE F(X,EXTRA1,EXTRA2,
C          EXTRA3,NEXTRA) WHICH MUST BE DECLARED EXTERNAL IN THE
C          ROUTINE THAT CALLS GAUS2.  THE VARIABLE OF INTEGRATION IS THE
C          FIRST ARGUMENT OF THE FUNCTION F.
C
C                          INPUT VARIABLES
```

```
C
C     F      EXTERNAL FUNCTION ROUTINE FOR INTEGRAND F(X,EXTRA1,EXTRA2,
C            EXTRA3,NEXTRA)
C     XL     LOWER LIMIT OF INTEGRATION (REAL)
C     XU     UPPER LIMIT OF INTEGRATION (REAL)
C     RELER  RELATIVE ERROR TOLERANCE (REAL)
C     ABSER  ABSOLUTE ERROR TOLERANCE (REAL)
C     EXTRA1 VARIABLE WHICH MAY BE PASSED TO FUNCTION F  (REAL)
C     EXTRA2 VARIABLE WHICH MAY BE PASSED TO FUNCTION F  (REAL)
C     EXTRA3 VARIABLE WHICH MAY BE PASSED TO FUNCTION F  (REAL)
C     NEXTRA VARIABLE WHICH MAY BE PASSED TO FUNCTION F (INTEGER)
C     IER    NORMALLY SET TO ZERO, BUT MAY BE SET TO 1 FOR THE
C             INTEGRAL TO BE COMPUTED BY A SINGLE APPLICATION
C             OF GAUSS-LEGENDRE FORMULA IF(10.*ABS(XU-XL)/RELER.LT.
C             AMAX1(ABS(XU),ABS(XL))          (INTEGER)
C
C           IF A1 AND A2 ARE THE  INTEGRALS COMPUTED ONCE OVER
C           THE REGION AND BY SUMMING THE VALUES IN BOTH HALVES
C           RESPECTIVELY,THEN CONVERGENCE IS OBTAINED WHEN
C               ABS(A1-A2)/RELER.LT.ABS(A2)+ABSER
C
C     ROUND MACHINE UNIT ROUND-OFF ERROR (I.E. THE SMALLEST NUMBER
C           ADDED TO 1.0 WHICH IS GREATER THAN 1.0)
C
C           MACHINES                       VALUES FOR ROUND
C           DG ECLIPSE                        1.2E-7
C           IBM 360/370                       9.6E-7
C           DEC 10                            7.7E-9
C           CDC 6600/7600                     7.7E-15
C           UNIVAC 1108                       1.5E-8
C
C
C                   OUTPUT VARIABLES
C
C     XL    UNCHANGED FROM INPUT FOR IER.LT.1. IF IER.GE.1, THEN EQUAL
C           TO LOWER LIMIT OF REGION FOR WHICH CONVERGENCE
C           WAS NOT OBTAINED
C     XU    UNCHANGED FROM INPUT FOR IER.LT.1. IF IER.GE.1, THEN EQUAL
C           TO UPPER LIMIT OF REGION FOR WHICH CONVERGENCE
C           WAS NOT OBTAINED
C     RELER UNCHANGED FROM INPUT UNLESS IER.GE.1, THEN EQUAL TO
C           INTEGRAL IN REGION FROM XL TO XU AT LAST ITERATION
C     ABSER UNCHANGED FROM INPUT UNLESS IER.GE.1. THEN EQUAL TO INTEGRAL
C           IN REGION FROM XL TO XU AT NEXT TO LAST ITERATION
C     ROUND UNCHANGED FROM INPUT
C     ANSWR VALUE OF INTEGRAL UNLESS IER.NE.0
C     IER   INTEGER ERROR FLAG
C             0 NO ERRORS, CONVERGENCE OBTAINED
C            -2 INTEGRATION DOMAIN IS TOO SMALL. ANSWR COMPUTED BY
C               SINGLE APPLICATION OF GAUSS-LEGENDRE FORMULA
C            -1 INTEGRATION DOMAIN IS TOO SMALL FOR GIVEN MACHINE
C               ROUND-OFF ERROR. ANSWR COMPUTED BY SINGLE APPLICATION
C               OF GAUSS-LEGENDRE FORMULA
```

```
C          .GE.1 NUMBER OF TIMES DIVIDED INTO HALVES BEFORE
C                REACHING MAXIMUM NUMBER OF SUBDIVISIONS. ANSWR
C                DETERMINED BY SINGLE APPLICATION OF GAUSS-LEGENDRE
C                FORMULA
C
C                            DIMENSIONS
C
C     TO RESET DEFAULT MAXIMUM NUMBER OF DIVISIONS (I.E. 20), CHANGE
C     NMAX TO THE NEW MAXIMUM PLUS 1. THE ARRAY DIMENSIONS SHOULD BE
C     A(2,NMAX),X(NMAX),Y(NMAX),H(NMAX) AND ISECT(NMAX)
C
C                         VARIABLES IN CODE
C
C     A(I,N)    INTEGRAL IN LEFT HALF (CORRESPONDING TO I=1), OR RIGHT
C               HALF (CORRESPONDING TO I=2) AT THE N-TH LEVEL.  FOR
C               N=1, INTEGRAL IS CONTAINED IN A(2,1) AND A(1,1)
C               IS NEVER USED
C     H(N)      STEP SIZE AT N-TH LEVEL
C     ISIDE(N)  SIDE AT N-TH LEVEL WHERE N=1 OR 2 CORRESPONDING TO
C               THE LEFT OR RIGHT HALF, RESPECTIVELY
C     N         LEVEL OF REGION
C     NMAX      MAXIMUM NUMBER OF LEVELS
C     X(N)      SMALLEST X VALUE AT THE N-TH LEVEL
C
      DIMENSION A(2,21),X(21),H(21),ISIDE(21)
      FUN(XD,HD)=0.5*HD*(F(XD+.2113248654052*HD,EXTRA1,EXTRA2,EXTRA3,
     $NEXTRA)+F(XD+.788675134598*HD,EXTRA1,EXTRA2,EXTRA3,NEXTRA))
      NMAX=21
C
      H(1)=XU-XL
      A(2,1)=FUN(XL,H(1))
      IF(IER.NE.1)GO TO 2
      IF(10.*ABS(H(1))/RELER.LT.AMAX1(ABS(XU),ABS(XL)))GO TO 7
C
C     CHECK THAT THE SIZE DOMAIN IS NOT TOO SMALL
C
    2 IF(ABS(XU-XL).GT.4.*ROUND*AMAX1(ABS(XL),ABS(XU)))GO TO 8
      ANSWR=A(2,1)
      IER=-2
      RETURN
C
C     DETERMINE THE MAXIMUM NUMBER OF SUBDIVISIONS BEFORE ROUND OFF
C     ERROR WOULD MAKE IT DIFFICULT TO DISTINGUISH POINTS IN THE DOMAIN
C
    8 RATIO=AMAX1(ABS(XU/H(1)),ABS(XL/H(1)))
      N1=-IFIX(1.4427*ALOG(RATIO*ROUND))
C+    N1=2-IFIX(1.4427*ALOG(RATIO*ROUND))
      NMAX=MINO(NMAX,N1)
      IF(NMAX.GT.1)GO TO 10
      IER=-1
      RETURN
C
   10 ISIDE(1)=2
```

```
      DO 1 I=2,NMAX
      ISIDE(I)=2
    1 H(I)=.5*H(I-1)
C
      X(2)=XL
      N=2
C
C
C     CALCULATE INTEGRAL IN EACH HALF.  AT LEVEL N, STORE RIGHT HALF
C     IN A(1,N) AND LEFT HALF IN A(2,N)
C
    4 SUM=0.
      A(1,N)=FUN(X(N),H(N))
      A(2,N)=FUN(X(N)+H(N),H(N))
      SUM=A(1,N)+A(2,N)
C
C     CHECK IF SUM IS EQUAL (WITHIN SPECIFIED TOLERANCES), TO THE
C     INTEGRAL COMPUTED OVER THE ENTIRE REGION. IF CONVERGENCE HAS NOT
C     BEEN OBTAINED, CHECK IF THE MAXIMUM NUMBER OF SUBDIVISIONS HAS
C     BEEN REACHED. IF THE MAXIMUM HAS NOT BEEN REACHED, RESET
C     THE LOWEST X VALUE AND SET ISIDE(N)=1 INDICATING A
C     NEW LEVEL AND RESTART BY COMPUTING THE INTEGRAL IN
C     THE LEFT HALF.
C
      IF(ABS(SUM-A(ISIDE(N),N-1))/RELER.LT.ABS(SUM)+ABSER)GO TO 3
      IF(N.EQ.NMAX)GO TO 9
      N=N+1
      ISIDE(N)=1
      X(N)=X(N-1)
      GO TO 4
C
C     NOW THAT CONVERGENCE HAS BEEN OBTAINED, REPLACE FIRST
C     APPROXIMATION OVER THE DOMAIN WITH SUM AND CHECK IF THIS
C     COMPLETES BOTH HALVES AT THE N-TH LEVEL (I.E. CHECK
C     IF ISIDE(N)=2). IF WE HAVE GONE THROUGH ALL REGIONS (I.E.N=2),
C     EXIT. IF ADDITIONAL LEVELS ARE TO BE COMPUTED (N.GT.2), REPLACE
C     FIRST APPROXIMATION WITH SUM AND MOVE TO A HIGHER LEVEL,
C     (I.E. A LOWER VALUE OF N).
C
    3 A(ISIDE(N),N-1)=SUM
      IF(ISIDE(N).EQ.1)GO TO 5
    6 IF(N.EQ.2)GO TO 7
      N=N-1
      A(ISIDE(N),N-1)=A(1,N)+A(2,N)
      IF(ISIDE(N).EQ.2)GO TO 6
C
C     MOVE LOWER LIMIT OF DOMAIN TO RIGHT HALF
C
    5 ISIDE(N)=2
      X(N)=X(N-1)+H(N-1)
      GO TO 4
C
C     TOO MANY ITERATIONS, SET ERROR FLAG
```

```
C
    9 IER=N-1
      XL=X(N)
      XU=X(N)+2.*H(N)
      RELER=SUM
      ABSER=A(ISIDE(N),N-1)
      RETURN
C
C     CONVERGENCE OBTAINED
C
    7 IER=0
      ANSWR=A(2,1)
D       IF (ANSWR.EQ.0.) WRITE(1,90) XL,XU,RELER
D 90    FORMAT(' GAUS2)    XL=',1PG15.7,5X,'XU=',G15.7,5X,
D     $'RELER=',G10.3)
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE GAUSBT(F,XL,XU,RELER,ABSER,ROUND,ANSWR,IER,IPRNT,
     $FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Outer Sectional Integral for Sectional
C        Coagulation Coefficients.
C
C  ON ENTRY:
C       F                  Function to be integrated
C       XL                 Lower Limit on Outer Integral
C       XU                 Upper Limit on Outer Integral
C       RELER              Relative Error Tolerance for Integration
C       ABSER              Absolute Error Tolerance for Integration
C       ROUND              Unit Round-Off Error (largest X that 1.+X=1.)
C       IPRNT              Logical Unit Number for Output Messages
C       FIXSZ              Inner Integral Size Limit
C       BASESZ             Inner Integral Size Limit
C       INNER              Flag to Interpret Inner Size Limits
C       TGAS               Gas Temperature [K]
C       PGAS               Gas Total Pressure [Pa]
C       NBTYPE             Flag for Type of Sectional Integral
C
C  ON RETURN:
C       ANSWR              Double Integral Value
C       IER                Error Return Flag
C
C  COMMENTS:
C       ALSO SEE DOCUMENTATION FOR GAUS2.
C
C***********************************************************************
C
```

```
      DIMENSION A(2,21),X(21),H(21),ISIDE(21)
      FUN(XD,HD)=0.5*HD*(F(XD+.2113248654052*HD,RELER,ABSER,ROUND,
     $ IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE)+
     $                        F(XD+.788675134598*HD,RELER,ABSER,ROUND,
     $ IPRNT,FIXSZ,BASESZ,INNER,TGAS,PGAS,NBTYPE))
      NMAX=21
      H(1)=XU-XL
      A(2,1)=FUN(XL,H(1))
      IF(IER.NE.1)GO TO 2
      IF(10.*ABS(H(1))/RELER.LT.AMAX1(ABS(XU),ABS(XL)))GO TO 7
    2 IF(ABS(XU-XL).GT.4.*ROUND*AMAX1(ABS(XL),ABS(XU)))GO TO 8
      ANSWR=A(2,1)
      IER=-2
      RETURN
    8 RATIO=AMAX1(ABS(XU/H(1)),ABS(XL/H(1)))
C+        N1=2-IFIX(1.4427*ALOG(RATIO*ROUND))
      N1=-IFIX(1.4427*ALOG(RATIO*ROUND))
      NMAX=MIN0(NMAX,N1)
      IF(NMAX.GT.1)GO TO 10
      IER=-1
      RETURN
   10 ISIDE(1)=2
      DO 1 I=2,NMAX
      ISIDE(I)=2
    1 H(I)=.5*H(I-1)
      X(2)=XL
      N=2
    4 SUM=0.
      A(1,N)=FUN(X(N),H(N))
      A(2,N)=FUN(X(N)+H(N),H(N))
      SUM=A(1,N)+A(2,N)
      IF(ABS(SUM-A(ISIDE(N),N-1))/RELER.LT.ABS(SUM)+ABSER)GO TO 3
      IF(N.EQ.NMAX)GO TO 9
      N=N+1
      ISIDE(N)=1
      X(N)=X(N-1)
      GO TO 4
    3 A(ISIDE(N),N-1)=SUM
      IF(ISIDE(N).EQ.1)GO TO 5
    6 IF(N.EQ.2)GO TO 7
      N=N-1
      A(ISIDE(N),N-1)=A(1,N)+A(2,N)
      IF(ISIDE(N).EQ.2)GO TO 6
    5 ISIDE(N)=2
      X(N)=X(N-1)+H(N-1)
      GO TO 4
    9 IER=N-1
      XL=X(N)
      XU=X(N)+2.*H(N)
      RELER=SUM
      ABSER=A(ISIDE(N),N-1)
      RETURN
    7 IER=0
```

```
      ANSWR=A(2,1)
D       IF (ANSWR.EQ.0.) WRITE(1,90) XL,XU,RELER
D 90    FORMAT(' GAUSBT)   XL=',1PG15.7,5X,'XU=',G15.7,5X,
D     $'RELER=',G10.3)
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      FUNCTION GROWTH(X,DUMMY,TGAS,PGAS,NBTYPE)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Condensational Growth Rate of a Particle
C
C  ON ENTRY:
C       X                 Log of Particle Mass [ln(kg)]
C       DUMMY             Not used
C       TGAS              Gas Temperature [K]
C       PGAS              Gas Total Pressure [Pa]
C       NBTYPE            Flag for coefficient type
C       /CONDNS/DIFFUS    Condensing Species Vapor Diffusivity [m**2/sec]
C       /CONDNS/CONMW     Molecular Weight of Condensing Molecule
C       /STOKES/DENSTY    Density of Liquid Condensing [kg/cu.m]
C       /GAS/PSAT         Saturation Pressure Of Condensing Species [Pa]
C
C  ON RETURN:
C       GROWTH            Particle Growth Rate, first-order in mass [/sec]
C                         = 6.28*DIFFUS*D*PSAT*DELSAT*WTCONM*F/(BOLTZ*TGAS*V)
C
C  COMMENTS:
C       None.
C
C***********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )       ! MKS
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
C       COMMON for Control Flags        AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
C
      VEL=SQRT(8.*RGAS*TGAS/(PI*CONMW)) ! Mean Kinetic Velocity, Monomer
      IF (USEBCE) THEN
        FREEMN=DIFFUS/VEL/BCE            ! True Mean Free Path, Monomer
      ELSE
        FREEMN=3.*DIFFUS/VEL             ! Adjusted Mean Free Path, Monomer
      END IF
C
C       Fuchs-Sutugin converges for large Kn to the kinetic limit only
C       if DIFFUS/(FREEMN*VEL)=1/3, a relationship obtained for
C       self-diffusion.  The assumed mean free path is intermediate
C       between that of the air and that of the condensing species,
C       and this method was first used by Pesty, Flagan, & Seinfeld
C       in Journal of Colloid and Interface Science, 91 (Feb. 1983), p.525
C
      V=EXP(X)                      ! Mass of Single Particle [kg]
      D=ZERO                        ! Initialize So ...
      CALL RHODD(V,D,RHO)           !  RHODD Returns Diameter, given Mass
C
      AKN=2.*FREEMN/D               ! Knudsen Number, Particle with Monomer
C
      IF (USEBCE) THEN
        F=FDCE(AKN,BCE)             ! Chapmann-Enskog
      ELSE
        F=FDFS(AKN)                 ! Modified Fuchs-Sututgin
      END IF
C
C       F goes to 1 for continuum (diffusive) regime aerosol with small Kn
C       F goes to 3/4 1/Kn for free molecule aerosol with large Kn
C
      GROWTH=12.*DIFFUS*PSAT*DELSAT*F*CONMW/(RGAS*TGAS*DENSTY*D*D)
C
C Or        GROWTH=2.*PI*PSAT*DELSAT*F*(DIFFUS*D/V)*(CONMW/RGAS)/TGAS
C
      RETURN
      END
C
C
C       This program uses either the modified Fuchs-Sutugin theory
C       originally used in MAEROS or the Chapmann and Enskog
C       theory used in Pete McMurry's models.  The latter theory
C       is probably superior.  Either model converges properly
C       to the kinetic and diffusive limits.  In the transition
C       region they may very by order of 10% (altho more like
C       1% if the BCE dimensionless group is actually 1/3).
C       The modified Fuchs-Sutugin model is empirical in nature.
C
```

```
C        BCE == Diffusivity / ( Mean Free Path * Monomer Mean Velocity)
C        For simple self diffusion, the above group should equal 1/3,
C        and this result is incorporated in the original Fuchs & Sutugin
C        flux matching approach for the transition regime.
C        Note Mean Free Path is for monomer, not for air.
C

         FUNCTION FDFS(KN)       ! Fuchs & Sutugin scaled to Diffusive Limit
         REAL KN
         FDFS=(1.+KN)/(1.+1.71*KN+1.333*KN*KN)
         RETURN
         END

         FUNCTION FKFS(KN)       ! Fuchs & Sutugin scaled to Kinetic Limit
         REAL KN
         FKFS = ( 1.333*KN * (1.+KN) ) / ( 1. + 1.71*KN + 1.333*KN*KN )
         RETURN
         END

         FUNCTION FKCE(KN,BCE)   ! Chapmann & Enskog scaled to Kinetic Limit
C        Note BCE=DGROUP=(diffus/lambda/vel1)
         REAL KN
         T1 = 4. * (BCE * KN) ** 2
         T2 = 2.88 * BCE * BCE * KN
         FKCE = ( T1 + T2 ) / ( T1 + T2 + 0.52*BCE*KN + 0.72*BCE )
         RETURN
         END

         FUNCTION FDCE(KN,BCE)   ! Chapmann & Enskog scaled to Continuum Limit
         REAL KN
         FDCE = (0.72+KN) / (0.72+0.52*KN+2.88*BCE*KN+4.*BCE*KN*KN)
         RETURN
         END
C
C----------------------------------------------------------------------
C
       SUBROUTINE J(S,RNJ,CRATE) ! CLASSICAL NUCLEATION Version
C
C**********************************************************************
C
C  PURPOSE:
C        To Calculate the CLASSICAL Homogeneous Nucleation Rate.
C
C  ON ENTRY:
C        S              Saturation Ratio [-]
C        CRATE          Condensation Rate onto Aerosol (without Kelvin
C                        effect included); Not Used Here [ug/cu.m/sec]
C        /NUCLO/...     Preset
C        /NUCL1/...     Preset
C
C  ON RETURN:
C        RNJ            Number Rate of Nucleation [#/cc/sec]
C
```

```
C  COMMENTS:
C       This routine simply calculates the classical rate of
C        homogenous nucleation, in cgs units.  Other nucleation
C        routines may be used (which is why CRATE is passed).
C
C***********************************************************************
C
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. , THREE=3. )
      PARAMETER ( TH1=ONE/THREE , TH2=TWO/THREE )
C
      PARAMETER ( RGAS = 8.31433E+7 )   ! Gas Constant, erg/K/mole
      PARAMETER ( BK = 1.38054E-16 )    ! Boltzmann Constant, erg/K/molecule
      PARAMETER ( AN = 6.02252E+23 )    ! Avogadro's Number, molecules/mole
C
      LOGICAL DEBUG /.FALSE./
      REAL MW                       ! Molecular Weight
C
C***            Nucleation COMMON Blocks
C
      COMMON /NUCL0/ T,VP,MW,DENSTY,SURTEN,RMS
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT
      COMMON /NUCL2/ VL,VM,DIAM,SAM,CS,VELQ,VPAT,DSMIN,DIKELV
      COMMON /NUCL3/ SR,GCRIT,DIMAA,BETAS,NFLAG,TN,RMNU,SRATE,RMNMIN
C
C***            Note RMNU is not set by JCLASS
C
      SR=S
      IF (S.LE.ONE) THEN
        RNJ=ZERO                    ! No Nucleation if SR <= 1
        RETURN
      ENDIF
C
      CON=S*CS                      ! Concentration, molecules/cc
      BETA=CON*VELQ                 ! Beta, surface collision rate, #/(cm*cm*sec)
      BETAS=BETA*SAM                ! Collision Frequency monomer-monomer /sec/1me

C  Note BETAS scales with g**(2/3) to become frequency a gmer is hit by monome

C  Actual gmer-monomer collision frequency is (1+1/g)**0.5 (1+g**1/3)**2 BETAS
C
C       DIMST=SURTEN*(VM**TH2)/(BK*T)     ! Surface:Thermal Energies
C        Note Monomer Surface Area = (36.*PI)**(1/3) * VM**(2/3)  assumed
C        Hence also  SUE = (36.*PI)**TH1 * DIMST = 4.83597586 DIMST
C
      SUE23= TH2 * SUE             ! = (32.*PI/3.)**TH1 * DIMST = 2/3 SUE
      GCRIT= ( SUE23 / ALOG(S) ) ** 3   ! g*: # in Critical Cluster
      CRITD=4.*SURTEN*VL/(RGAS*T*ALOG(S))        ! Critical Diameter, cm
C
      WCR=0.5*(SUE23**3)/(ALOG(S))**2
      ZELD=SQRT(WCR/(3.*PI))/GCRIT      ! From Hirth & Pound - correct
      RNJ=ZELD*BETAS*(GCRIT**TH2)*CON*EXP(-WCR)
      TN=1./(4.*BETAS*GCRIT**TH2*ZELD**2)        ! Collins Nucl Time Lag
      RNJFR=RSCALE*SQRT(PI/6.*ALOG(S)*GCRIT**TH1)*S**(TWO-GCRIT/TWO)
C
C        Equivalent Classical Nucleation Expression
C       RNJ2=(S*CS)**2*SAM/PI*SQRT(RGAS*T*SUE/18./MW)*EXP(-WCR)
C       TYPE *,RNJ,RNJ2
C
C-----------------------------------------------------------------------
    1 FORMAT()
    2 FORMAT(/)
        IF (DEBUG) THEN
      WRITE(21,105) 'ORGANIC',T
  105 FORMAT(' Nucleating Species is ',A8,' at',F6.1,' K.')
      WRITE(21,110) VPAT,S
  110 FORMAT(' Vapor pressure is ',1PE9.2,' Atm with a ',
     $ 'Saturation Ratio of ',0PF10.4)
      WRITE(21,115) SURTEN,SUE
  115 FORMAT(' Surface Tension',F7.2,' dynes/cm',6X,'Monomer ',
     $ 'Surface Energy',F7.3,' kT')
      WRITE(21,120) RSCALE,CS
  120 FORMAT(' Collisional RSCALE is',1PE11.3,' #/cc/sec with Ns at',
     $ E11.3,' #/cc')
      WRITE(21,125) RMS,DIMSOR
  125 FORMAT(' Source Rate is',1PE11.3,' ug/cu.m./sec, or',
     $ E11.3,' Nondimensionally')
      WRITE(21,130) TB,TS
  130 FORMAT(' Time Constants (Seconds):  Collision',1PE11.3,5X,
     $ 'Source',1PE11.3)
      WRITE(21,131) TN
  131 FORMAT(' Nucleation Time Lag is',1PE11.3,' Seconds')
       IF (GCRIT.LT.1.E6) THEN
      WRITE(21,135) CRITD,GCRIT
  135 FORMAT(' A',8PF9.2,' Angstrom Critical Cluster contains',
     $ 0PF13.2,' molecules')
       ELSE
      WRITE(21,136) CRITD,GCRIT
  136 FORMAT(' A',8PF9.2,' Angstrom Critical Cluster contains',
     $ 1PE11.3,' molecules')
       ENDIF
      WRITE(21,140) WCR
  140 FORMAT(' Maximum Energy Barrier is ',3(F15.3,' kT',:))
      WRITE(21,145) ZELD
  145 FORMAT(' The Zeldovich Factor is ',1PE12.3)
      WRITE(21,150) RNJ
  150 FORMAT(' Classical B-D-Z Nucleation Rate: ',1PE12.3,' #/cc/sec')
      WRITE(21,151) RNJFR
  151 FORMAT(' Friedlanders Nucleation Rate: ',1PE12.3,' #/cc/sec')
C       TYPE 32, BETA
C   32 FORMAT(' Beta is ',1PE10.3,' collisions per second per sq cm')
C       TYPE 33, VELO
C   33 FORMAT(' The mean approach velocity of the monomer is',
C     $  1PE10.3,' cm/sec')
C       TYPE 34, BETAS
C   34 FORMAT(' A Monomer area has',1PE11.3,' impacts/second. (BETAS)')
```

```
C       TYPE 36, ONE/BETAS,ONE/RAMS
C    36 FORMAT(/' The monomer time constants are as follows (seconds):'
C      $ /' Cluster Growth',1PE11.3,6X,'Scavenging',E11.3)
C       TYPE 37, DIMCS
C    37 FORMAT(' The Dimensionless Cluster Scavenging Number is',
C      $ 1PE11.3/)
C       TYPE 39, RSCALE
C    39 FORMAT(' Characteristic Collision Rate RSCALE is',1PE11.3,
C      $ ' #/cc/sec total (sat.)')
C       TYPE 49, SUE23   ! BDZ takes exp(SUE23*approx), approx off by 1.5+
C    49 FORMAT(' Summation Prefactor in Exponential for N1/Eg:',1PE10.3/)
        DEBUG=.FALSE.      ! ONCE ONLY
          ENDIF
C-----------------------------------------------------------------------
          IF (RNJ.GT.1.E-5 .AND. DSMIN.LT.CRITD) THEN
        WRITE(21,900)
  900 FORMAT(' Warning: Nucleation with CRITD > DSMIN')
          ENDIF
        RETURN
        END
C
C-----------------------------------------------------------------------
C
        SUBROUTINE JSET(DMIN)
C
C***********************************************************************
C
C  PURPOSE:
C        To set up COMMON blocks for Nucleation Routine J (in cgs units).
C
C  ON ENTRY:
C        /NUCLO/ variables must be preset.
C
C  ON RETURN:
C        /NUCL1/, /NUCL2/ variables set.
C        /TRANS/ variables set.
C
C  COMMENTS:
C        BCE must be set elsewhere.
C        JSET should be called once before Nucleation routine J is called;
C         if conditions (T,VP,RMS,PGAS, etc.) change, recall JSET.
C
C***********************************************************************
C
        PARAMETER ( PI = 3.1415927 )
        PARAMETER ( ZERO=0. , ONE=1. , TWO=2. , THREE=3. )
        PARAMETER ( TH1=ONE/THREE , TH2=TWO/THREE )
C
        PARAMETER ( RGAS = 8.31433E+7 )    ! Gas Constant, erg/K/mole
        PARAMETER ( BK = 1.38054E-16 )     ! Boltzmann Constant, erg/K/molecule
        PARAMETER ( AN = 6.02252E+23 )     ! Avogadro's Number, molecules/mole
C
        REAL MW
```

```
      COMMON /NUCL0/ T,VP,MW,DENSTY,SURTEN,RMS,PGAS
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT
      COMMON /NUCL2/ VL,VM,DIAM,SAM,CS,VELQ,VPAT,DSMIN,DIKELV
      COMMON /TRANS/ DIFFUS,DIMDIM,BCE,AMFP,CMFP
C
      DSMIN=DMIN                    ! cm diameter of smallest aerosol
C
      WEIGHT=1.E12*MW/AN            ! ug/cu.m per #/cc monomer
      SOURCE=RMS/WEIGHT             ! Source rate in #/cc/sec
      IF (SOURCE.EQ.ZERO) SOURCE=-1.     ! Avoid /0 errors
      VL=MW/DENSTY                  ! Liquid Molar Volume, cc/mole
      VM=VL/AN                      ! Molecular Volume, cc/molecule
      DIAM=(6.*VM/PI)**THI          ! Molecular Diameter, cm
      SAM=PI*DIAM*DIAM              ! Molecular Surface Area, cm*cm
      CS=VP/(BK*T)                  ! Concentration (Sat.), molecules/cc
      VELQ=SQRT(RGAS*T/(TWO*PI*MW))      ! 0.25 Mean Molecular Velocity, cm/sec
      VPAT=VP/1.0133E+6             ! Vapor Pressure, atm
      SUE=SURTEN*SAM/BK/T           ! Surface Energy in kT units for monomer
      RSCALE=SAM*CS*CS*VELQ         ! Characteristic Rate Scale, #/cc/sec
      TB=CS/RSCALE                  ! Characteristic Collision Time, sec, sat.
      TS=CS/SOURCE                  ! Characteristic Source Time, seconds, sat.
      DIMSOR=SOURCE/RSCALE          ! Dimensionless Source Rate
      DIKELV=4.*SURTEN*VM/(BK*T)         ! Characteristic Kelvin Diameter
C
C       The following are used only by the cgs condensation rate routines
C
C                Collision Diameter based on BS&L recommendations
      CDAIR=3.617E-8                ! Collision Diameter of Air (BS&L) [cm]
      CDCON=0.98*DIAM               ! Collision Diameter of Condensible
      COLLDI=(CDCON+CDAIR)/2.       ! Collision Diameter (condensible with air)
      AIRN=PGAS/BK/T                ! Number Concentration of Air [molecules/cc]
      AMFP=1./(SQRT(2.)*AIRN*PI*CDAIR**2)        ! Air Mean Free Path
      CMFP=1./(SQRT(1.+MW/29.0)*AIRN*PI*COLLDI**2)       ! Condensible M.F.P.
C
C***             Estimate Diffusivity of Monomer in Air if unknown
C
C      IF (BCE.EQ.ZERO.AND.DIFFUS.EQ.ZERO)    DIFFUS=(2./3.)*
C    $      ((RGAS*T/PI)**1.5)*SQRT(0.5/MW+0.5/29.0)/PGAS/(COLLDI**2)/AN
C       The above is taken from illustrative simple theory of BS&L.
C
C       Use Chapman-Enskog theory for good estimate of diffusivity:
C       (re: Jim Davis, AS&T, 1983, as well as eq. 16.4-12 in BS&L)
C
      IF (BCE.EQ.ZERO.AND.DIFFUS.EQ.ZERO)  DIFFUS=(3./8./PI)*
     $      SQRT(PI*RGAS*T*(1./MW+1./29.0)/2.)/((COLLDI**2)*AIRN)
C
C       Note: The Chapman-Enskog DIFFUS is 1.767 of the simple BS&L
C       illustrative theory prediction (based on 0th order diffusion),
C       assuming a collision integral of unity (in denominator of CE).
C
C       Thus simple BS&L or Fuchs-Sutugin predicts BCE = (1+Z)/6.
C       But Chapmann-Enskog predicts BCE = (1+Z)*3*PI/32.
C       This latter expression is a very fast way to get BCE and
```

```
C       thus DIFFUS for this subroutine.
C
      IF (BCE.EQ.ZERO) BCE=DIFFUS/(4.*VELQ*CMFP)
      IF (DIFFUS.EQ.ZERO)  DIFFUS=4.*VELQ*CMFP*BCE
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE JMKS(S,RJM,CONRAT,GC)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate Homogeneous Nucleation Rate.  Merely interfaces the
C        MKS Sectional Aerosol Model with the cgs Nucleation Routine J.
C
C  ON ENTRY:
C       S        Saturation Ratio [-]
C       CONRAT  Mass Rate of Condensation (Omit Kelvin Effect!) [kg/cu.m/sec]
C       /NUCL#/ (selected) COMMON variables preset
C
C  ON RETURN:
C       RJM      Mass Rate of Homogeneous Nucleation [kg/cu.m/sec]
C       GC       Critical Number (REAL) [-]
C       /NUCL3/ variables updated
C
C  COMMENTS:
C       Called by DIFFUN.
C       Unless cluster scavenging effects are included, (steady state)
C        nucleation (number) rate is a function only of Saturation Ratio,
C        given T to fix the physical properties of the vapor species.
C       The mass rate of nucleation is defined based on the minimum
C        aerosol diameter, DIN, for the model.
C
C***********************************************************************
C
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)         ! Sectional Diam & Masses
C
      RJMIN=1.E9*RJM               ! ug/cu.m from kg/cu.m
      CRATE=1.E9*CONRAT            ! (Don't include Kelvin Effect for this)
C
      CALL J(S,RJN,CRATE)          ! Use cgs Homogeneous Nucleation Routine
C
      VHMEAN=ALOG(VS(2)/VS(1))/(1./VS(1)-1./VS(2))  ! kg mean particle
C
```

```
C        Use VHMEAN instead of VS(1) because it is number flux that
C        is essentially conserved by the nucleation process.
C        The exact size at which a nucleated particle begins is not
C        as important as the number of particles formed.
C
      RJM=RJN*(VHMEAN*1.E6)      ! #/cc/sec * kg/# * cc/cu.m = kg/cu.m/sec
      GC=GCRIT                   ! Critical Number
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE MAEROS(TIME,DELTIM,Q,TGAS,PGAS,IPRNT,IFLAG,NEWCOF)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate an Aerosol Size Distribution,
C        At a Future Time, Using a Sectional Representation.
C       This Routine is the Driver for the Expanded Sectional
C        MultiComponent Aerosol Package (ESMAP).
C
C  ON ENTRY (ARGUMENTS):
C       TIME             Current Time [sec]
C       DELTIM           Time Step, after which MAEROS returns [sec]
C       Q(NEQ)           Sectional Mass Array [kg/cu.m]
C       TGAS             Gas Temperature [K]
C       PGAS             Pressure, total [Pa]
C       IPRNT            Logical Unit Number for Output (often 6)
C       IFLAG            Flag for Integration Routine
C       NEWCOF           Flag that controls which coefficients are calculated;
C                         Negative values cause use of current coefficients,
C                         while Positive values call for the following action:
C               1 = Interpolate Temperature and Pressure        (4 sets)
C               2 = Only Use TGAS1 and PGAS1                     (1 set)
C               3 = Interpolate Temperature, Use PGAS1          (2 sets)
C               4 = Interpolate Pressure, Use TGAS1             (2 set)
C               5 = Recalculate Only Deposition Set(s)
C               6 = Recalculate Only Condensation Set(s)
C               7 = Recalculate Only Deposition & Condensation Set(s)
C               8 = Modify Condensation Coefficients by factor DELSAT
C               9 = Recalculate Only Condensation for TGAS1,PGAS1
C               11-15 = 1-5 respectively, but No Condensation
C
C  ON ENTRY (COMMON):
C       /TPSET/ TGAS1,TGAS2      Min and Max Temperatures [K]
C       ...     PGAS1,PGAS2      Min and Max Pressures [Pa]
C       /PSRATE/PSRATE(NEMAX)    Sectional Particle Source Rates [kg/cu.m/sec]
C       /DEPSIT/DEPSIT(3,KC)     Mass Deposited on (Surface,Component) [kg]
C       /ROUND/ UROUND           Machine Unit Round-Off Error
C       /INDEX/ MS,KC            Number of Size Sections and Components
C
```

```
C  ON RETURN (ARGUMENTS):
C        Q                    Sectional Mass Array has been updated.
C        TIME                 Updated to new Time.
C        NEWCOF               Set to Negative of Initial Absolute Value.
C
C  ON RETURN (COMMON):
C        /INDEX/
C
C
C  COMMENTS:
C        This version of MAEROS uses the EPISODE integration package
C        (Note Episode was modified to use higher IFLAG with a YMIN)
C
C        Program was revised by DALE WARREN to:
C         - couple a vapor phase concentration to aerosol condensation
C         - handle rapid condensation processes while conserving number
C         - handle homogeneous nucleation in the presence of an aerosol
C         - use microgram/cubic meter units in expanded printout
C         - reduce roundoff errors (often Fatal) for lower precision machines
C         - optionally use Jim Crump's unified container deposition model
C         - use data file storage of calculated average coefficients
C         - store more state variables and parameters in COMMON blocks
C         - use structured FORTRAN-77 for increased clarity and efficiency
C         - include more program comments (mine usually lower case)
C
C        This code is based on the MAEROS package written by Fred Gelbard,
C        and available from SANDIA LABORATORIES.
C
C  LOCAL VARIABLES:
C        NEWSET             Keeps track of how many T,P cases needed:
C                =1 (T1/T2,P1/P2)  =2 (T1,P1)  =3 (T1/T2,P1)  =4 (T1,P1/P2)
C
C***********************************************************************
C
C       THE MASS OF
C       EACH COMPONENT DEPOSITED IS ALSO CALCULATED BY
C       USING A MASS BALANCE TO DETERMINE THE MASS REMOVED
C       FROM THE AEROSOL AND PARTITIONING THAT MASS TO THE
C       THREE DEPOSITION SURFACES BASED ON THE RELATIVE
C       REMOVAL RATES ON THE SURFACES AVERAGED OVER THE
C       TIME STEP. THIS CODE WAS WRITTEN BY FRED GELBARD.
C
C     UROUND=MACHINE ROUND-OFF ERROR (I.E. SMALLEST NUMBER ADDED TO ONE
C            WHICH IS GREATER THAN ONE)
C             MACHINES                    VALUES FOR UROUND
C             DG ECLIPSE                     1.2E-7
C             IBM 360/370                    9.6E-7
C             DEC 10                         7.7E-9
C             CDC 6600/7600                  7.7E-15
C             UNIVAC 1108                    1.5E-8
C
C***********************************************************************
C
```

```
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                  ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )        ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )          ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )                ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )       ! MKS
      DIMENSION Q(NEMAX),QKSUM(8),QKLEFT(8),QT(MMAX)
      DIMENSION WORK(NWMAX),IWORK(5)     ! Workspace for Integration
D       DIMENSION DQDTJ(NEMAX)    ! Only needed to set /NUCL/ exactly -DRW
      COMMON /INDEX/  MS,KC,NQV,NQN,
     $  NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF     ! Pointers
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /TPSET/ TGAS1,TGAS2,PGAS1,PGAS2     ! T,P set for interpolation
      COMMON /AVGCOF/ COEFAV(NCMAX)       ! Sectional Coefficients
      COMMON /PSRATE/  PSRATE(NEMAX)      ! Sectional Particle Source Rates
      COMMON /DEPSIT/ DEPSIT(3,2)         ! Deposited Masses
C  DEPSIT array is 3 surfaces by KCOMP components.  Approximate values.
      COMMON /PARINT/ RELE,ABSE,KTOL,MFEPI,HO    ! Integration Parameters
      COMMON /EPCOMR/ NRMIN,NRMAX         ! COMMON for DRIVES in EPIS
      COMMON /DBLK/ CT1P1(NCMAX),CT1P2(NCMAX),CT2P1(NCMAX),CT2P2(NCMAX)
      DATA CT1P1,CT2P1 /NCMAX*0.,NCMAX*0./
      DATA CT1P2,CT2P2 /NCMAX*0.,NCMAX*0./
      DATA NRMIN  / 1 /           ! First Q that must stay non-negative
      DATA TNMASS / ZERO /        ! Total (cumulative) Negative Mass
      DATA NEWSET / 0 /           ! Number of T,P sets
      EXTERNAL DIFFUN                       ! Derivative Calculator
C
C***             CHECK IF VARIABLES HAVE ACCEPTABLE VALUES
C
      CALL CHECKE(TIME,DELTIM,Q,TGAS,PGAS,IPRNT,IFLAG,NEWCOF)
      IF (IFLAG.LT.-1 .OR. IFLAG.GT.3) THEN
        IF (IFLAG.NE.7) THEN                ! New EPIEXP
          WRITE(IPRNT,31) IFLAG,TIME     ! Bad Input from Main Program
          STOP 'SUBROUTINE CHECKE DETECTED DATA PROBLEM'
        END IF
      END IF
   31 FORMAT(' CHECK RETURNED ERROR CODE',I4,'  AT TIME =',1PE15.4)
C
      MF=MFEPI              ! Main Program Sets Method Flag for Episode
      AERROR=RELE           ! Main Sets (Relative) Local Error Tolerance
      NQMK=MS*KC                            ! Number of aerosol Q sections
      NQV=NQMK+1                            ! Allow for one vapor phase D.E.
      NQN=NQMK+2                            ! Follow Total Nucleation
      NEQ=NQN                              ! Number of Simultaneous O.D.E.s
      NRMAX=NQN                            ! Last Q that must stay non-negative
C
C***             SET THE CONDENSATION FLAG
C
      IF (IABS(NEWCOF).GE.11) THEN
        ICONDN=0                           ! No Condensation
      ELSE
        ICONDN=1                           ! Need Condensation Coefficients
      END IF
C
      IF (KTOL.LE.5.AND.IFLAG.EQ.7) IFLAG=-1 ! If using EPI.EXP have shortcut
C
C***             SET /INDEX/ POINTERS TO THE COEFFICIENT ARRAY, COEFAV
C
      IF (IFLAG.GE.0) THEN
        NB2A=((MS-2)*(MS-1))/2
        NB2B=((MS-1)*MS)/2+NB2A
        NB3=NB2B+((MS-1)*MS)/2
        NB4=NB3+MS
        NDEPST=NB4+((MS-1)*MS)/2           ! Offset for Deposition Coef.
        NGROW=NDEPST+3*MS                  ! Offset for Growth Coef.
        NUMCOF=NGROW+ICONDN*(3*MS-1)       ! If cond, NUMCOF= 2*MS*MS + 4*MS
      END IF
      IF (IFLAG.EQ.-1) IFLAG=1
C
C***             COMPUTE COEFFICIENTS AS SPECIFIED BY NEWCOF
C
      IF (NEWCOF.GE.0.AND.NEWCOF.NE.8) THEN      ! Need to Do Integrals
C
        IF (NEWCOF.LT.5) THEN              ! Try to set NEWSET for (T,P) range
          NEWSET=NEWCOF
        ELSE IF (NEWCOF.GE.11.AND.NEWCOF.LE.14) THEN
          NEWSET=NEWCOF-10
        ELSE IF (NEWCOF.EQ.9) THEN
          NEWSET=2
        END IF
C
        IF (NEWCOF.EQ.5.OR.NEWCOF.EQ.7.OR.NEWCOF.EQ.15) THEN
          ISTART=NDEPST+1
        ELSE IF (NEWCOF.EQ.6) THEN
          ISTART=NGROW+1
        ELSE
          ISTART=1
        END IF                                   ! ISTART SET
C
        IF (NEWCOF.EQ.5.OR.NEWCOF.GE.11) THEN
          IFNSH=NGROW
        ELSE
          IFNSH=NUMCOF
        END IF                                   ! IFNSH SET
C
        CALL COEF(NEWCOF,TGAS1,PGAS1,IPRNT)
```

```
C
        DO I=ISTART,IFNSH                               ! Transfer to CT1P1
          CT1P1(I)=COEFAV(I)
        END DO
C
        IF (NEWSET.EQ.1 .OR. NEWSET.EQ.3) THEN
          CALL COEF(NEWCOF,TGAS2,PGAS1,IPRNT)
          DO I=ISTART,IFNSH
            CT2P1(I)=COEFAV(I)                          ! Transfer to CT2P1
          END DO
        END IF
C
        IF (NEWSET.EQ.1 .OR. NEWSET.EQ.4) THEN
          CALL COEF(NEWCOF,TGAS1,PGAS2,IPRNT)
          DO I=ISTART,IFNSH
            CT1P2(I)=COEFAV(I)                          ! Transfer to CT1P2
          END DO
        END IF
C
        IF (NEWSET.EQ.1) THEN
          CALL COEF(NEWCOF,TGAS2,PGAS2,IPRNT)
          DO I=ISTART,IFNSH
            CT2P2(I)=COEFAV(I)                          ! Transfer to CT2P2
          END DO
        END IF
      END IF              ! CT#P# arrays set as required by NEWCOF, NEWSET
C
C***            SET ACTIVE COEFAV ARRAY OF COEFFICIENTS
C
      IF (TGAS.EQ.TGAS1 .AND. PGAS.EQ.PGAS1) THEN
        DO I=1,NUMCOF
          COEFAV(I)=CT1P1(I)
        END DO
      ELSE      ! Linear Interpolation of Available Coefficients in T,P
        TZ=(TGAS-TGAS1)/(TGAS2-TGAS1)
        PZ=(PGAS-PGAS1)/(PGAS2-PGAS1)
        DO I=1,NUMCOF
          COEFAV(I) = (1.-TZ) * ( (1.-PZ)*CT1P1(I) + PZ*CT1P2(I) )
     $              +    TZ   * ( (1.-PZ)*CT2P1(I) + PZ*CT2P2(I) )
        END DO
      END IF
C
C***            SET GAS PROPERTIES (IN /GAS/ COMMON) TO CURRENT VALUES
C
      CALL SETGAS(TGAS,PGAS)    ! Set TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS
C
C***
C
      IF (NEWCOF.EQ.8) THEN
        ISTART=NGROW+1
        DO I=ISTART,NUMCOF
          COEFAV(I)=DELSAT*COEFAV(I)
        END DO
      END IF
C
      NEWCOF=-IABS(NEWCOF)       ! Set Negative As Have Desired COEFAV
C
C     STORE THE INITIAL DEPOSITION RATES (IN KG/SEC) OF THE K-TH
C     COMPONENT ON THE J-TH DEPOSITION SURFACE IN DEPSIT(J,K)
C
      DO J=1,3
        DO K=1,KC
          DEPSIT(J,K)=ZERO
          DO L=1,MS
      DEPSIT(J,K)=DEPSIT(J,K)+COEFAV(3*(L-1)+NDEPST+J)*Q(K+(L-1)*KC)
          END DO
        END DO
      END DO
C
C     STORE THE AEROSOL RELEASED OVER THE TIME STEP (IN KG), AND THE
C     INITIAL SUSPENDED OF THE K-TH COMPONENT IN QKSUM(K)
C
      DO K=1,KC
        SORSK=ZERO
        QKSUM(K)=ZERO
        DO L=1,MS
          SORSK=SORSK+PSRATE((L-1)*KC+K)
          QKSUM(K)=QKSUM(K)+Q((L-1)*KC+K)
        END DO
        QKSUM(K)=(QKSUM(K)+SORSK*DELTIM)*VOLUME
      END DO
C
C***             STORE THE INITIAL CONDENSATION RATE (of KC) IN CONDNS
C
      IF (ICONDN.NE.0) THEN              ! Condensation
        DO L=1,MS
          SUM=ZERO
          DO K=1,KC
            SUM=SUM+Q(KC*(L-1)+K)
          END DO
          QT(L)=SUM
        END DO
        QVAP=Q(NQV)                      ! Must set for CALCON
        CALL CALCON(QT,QVAP,SR,CONDNS,Z)         ! Find CONDNS
      END IF
C
C***             CALL THE TIME INTEGRATION PACKAGE TO TAKE A TIME STEP
C
      TOUT=TIME+DELTIM                   ! Destination Time
C
   70 WRITE(3,235) NEQ,TIME,HO,TOUT,AERROR,ABSE,KTOL,MF,IFLAG
  235 FORMAT(/5X,'ON CALL TO DRIVES:'/' NEQN=',I3,3X,'TIME=',1PE9.2,
     $ 3X,'STEP=',E10.2,3X,'TOUT=',E9.2/ ' RELE=',E9.2,4X
     $ 'ABSE=',E9.2,4X,'KTOL=',I4,4X,'MF=',I4,4X,'IFLAG=',I4/)
C
      CALL DRIVE(NEQ,TIME,HO,Q,TOUT,AERROR,KTOL,MF,IFLAG)
```

```
C
C*       CALL RKF45(DIFFUN,NEQ,Q,TIME,TOUT,RELE,ABSE,IFLAG,
C*      $ WORK,IWORK,UROUND)
C
C     IF THE CONCENTRATION OF A COMPONENT GOES NEGATIVE, SET IT TO
C     ZERO AND RESET IFLAG TO -1 TO RESTART TIME INTEGRATION
C
      IF (IFLAG.EQ.0 .OR. IFLAG.EQ.-7) THEN      ! No serious error
        NERRS=0
        QNMASS=ZERO
        DO I=1,NQMK
          IF (Q(I).LT.ZERO) THEN
            INDY=7                 ! With RK, IFLAG=-1
            NERRS=NERRS+1          ! Keep track of number of negatives
            QNMASS=QNMASS-Q(I)     ! Negative Mass this DELTIM time period
            Q(I)=ZERO              ! Correct negative mass to zero
          END IF
        END DO
        TNMASS=TNMASS+QNMASS
        IF (NERRS.GT.0) WRITE(4,840) TIME,NERRS,QNMASS*1.E9,TNMASS*1.E9
  840 FORMAT(/'  AT TIME',1PE10.3,' THERE WERE ',I3,
     $ ' NEGATIVE MASS SECTIONS FOUND'/
     $ ' NEGATIVE MASS ELIMINATED WAS',1P2E13.3,' UG/CU.M.'/)
C
        IF (IFLAG.EQ.-7) THEN      ! Reduce H0 step size and integrate on
          IFLAG=7                  ! Flag that Y is changed slightly
          DELTIM=TOUT-TIME
          H0=H0/10.                        ! Reduce Step Size
          GOTO 70                          ! Continue Integrating
        END IF
        IF (INDY.EQ.7) THEN
          IFLAG=7                  ! Must Start Again For Negative Mass
          H0=H0/10.                ! Reduce Step Size
        END IF
      END IF
C
      IF (ICONDN.NE.0) THEN
C
C     ADD THE FINAL CONDENSATION RATE (IN KG/SEC) OF THE LAST
C     COMPONENT TO THE INITIAL CONDENSATION RATE AND DIVIDE BY 2
C     TO OBTAIN THE AVERAGE CONDENSATION RATE OVER THE TIME STEP.
C     ADD THE CALCULATED AVERAGE CONDENSATION RATE TO QKSUM(KC)
C     TO OBTAIN THE TOTAL FORMATION RATE OF THE LAST COMPONENT IN
C     THE AEROSOL PHASE.
C
        DO L=1,MS          ! Find Total Mass in Each Size Section
          SUM=ZERO
          DO K=1,KC
            SUM=SUM+Q(KC*(L-1)+K)
          END DO
          QT(L)=SUM
        END DO
        QVAP=Q(NOV)                        ! Must set for CALCON
```

```
         CALL CALCON(QT,QVAP,SR,CONRAT,Z)
         CONDNS=0.5*(CONDNS+CONRAT)       ! Mean Condensation Rate in Interval
         QKSUM(KC)=QKSUM(KC)+CONDNS*VOLUME*DELTIM          ! Add Condensed Mass
       END IF
C
C      SUBTRACT THE FINAL SUSPENDED MASS FROM THE INITIAL, ADDED AND
C      CONDENSED MASS OF THE K-TH COMPONENT AND STORE THAT IN
C      QKLEFT(K).  THEREFORE, BY A MASS BALANCE, QKLEFT(K) IS THE
C      DEPOSITED MASS OF THE K-TH COMPONENT FOR THE TIME STEP
C
C        This method of estimating the condensed and deposited mass
C        of each component is not very accurate when long time steps
C        are used and the aerosol changes noticeably, affecting the
C        condensation rate if a constant supersaturation is assumed.
C        However, if the condensation rate is fixed (or linear with time),
C        the method is as accurate as the alternative, which is
C        to include the amount of each component condensed and/or deposited
C        as additional differential equations for the RKF routines.
C
   21 DO K=1,KC
         QKLEFT(K)=ZERO
         DO L=1,MS
           QKLEFT(K)=QKLEFT(K)+Q((L-1)*KC+K)      ! Component K mass in aerosol
         END DO
         QKLEFT(K)=QKSUM(K)-QKLEFT(K)*VOLUME      ! Component K mass lost
         IF (QKLEFT(K).LT.ZERO) QKLEFT(K)=ZERO
       END DO
C
C      ADD THE FINAL DEPOSITION RATE ON THE J-TH DEPOSITION SURFACE
C      FOR THE K-TH COMPONENT TO DEPSIT(J,K) AND DIVIDE BY 2 TO
C      OBTAIN THE AVERAGE DEPOSITION RATE OF THE K-TH COMPONENT ON
C      ALL THREE DEPOSITION SURFACES IN QKSUM(K)
C
C        Note this does not account for nucleation
C
       DO K=1,KC
         QKSUM(K)=ZERO
         DO J=1,3
           SUM=ZERO
           DO L=1,MS
             SUM=SUM+COEFAV(3*(L-1)+NDEPST+J)*Q(K+(L-1)*KC)
           END DO
           DEPSIT(J,K)=0.5*(DEPSIT(J,K)+SUM)
           QKSUM(K)=QKSUM(K)+DEPSIT(J,K)
         END DO
       END DO
C
C      COMPUTE THE MASS DEPOSITED OF THE K-TH COMPONENT ON THE J-TH
C      SURFACE BY PARTITIONING THE TOTAL MASS DEPOSITED OF THE K-TH
C      COMPONENT (I.E. QKLEFT(K)), BASED ON THE WEIGHTED AVERAGE
C      DEPOSITION RATE (I.E. DEPSIT(J,K)/QKSUM(K))
C
       DO K=1,KC
```

```
          RATIO=ZERO
          IF (QKSUM(K).GT.ZERO) RATIO=QKLEFT(K)/QKSUM(K)
          DO J=1,3
            DEPSIT(J,K)=DEPSIT(J,K)*RATIO
          END DO
        END DO
C
D       CALL DIFFUN(NEQ,TIME,Q,DQDTJ)      ! Only to set /NUCL/ exactly -DRW
C
        IF (IFLAG.GE.0) THEN
          RETURN
        ELSE
          WRITE(IPRNT,27) IFLAG,TIME
   27 FORMAT(//' EPISODE ERROR NUMBER',I4,3X,'SEE EPISODE LISTING'/
     $   3X,'TIME REACHED WHEN ERROR OCCURED =',E11.4//)
          WRITE(IPRNT,29) (Q(I),I=1,NEQ)
   29 FORMAT(' VALUES OF Q ARRAY'/(1P8E10.2))
          RETURN
        END IF
C
        END
C
C-------------------------------------------------------------------
C
        SUBROUTINE NLIST(IO,IARG)
C
C*******************************************************************
C
C  PURPOSE:
C        SUBROUTINE TO LIST PROPERTIES OF CONDENSING SYSTEM.
C
C  ON ENTRY:
C        IO       File Number to Write Out To
C        IARG     Specifies extent of information to write:
C                 0 for Very Brief
C                 1 for Brief
C                 2 for Usual
C                 3 for Usual + SQN Flags
C                 4 for ALL
C        /NUCL#/ varibles preset
C
C  ON RETURN:
C        All unchanged.
C
C  COMMENTS:
C        None.
C
C*******************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )       ! MKS
      PARAMETER ( BK = 1.38054E-16 )      ! Boltzmann Constant, erg/K/molecule
      PARAMETER ( AN = 6.02252E+23 )     ! Avogadro's Number, molecules/mole
      REAL MW                          ! Molecular Weight
      COMMON /NUCL0/ T,VP,MW,DENSTY,SURTEN,RMS,PGAS
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT
      COMMON /NUCL2/ VL,VM,DIAM,SAM,CS,VELQ,VPAT,DSMIN,DIKELV
      COMMON /TRANS/ DIFFUS,DIMDIM,BCE,AMFP,CMFP
C        COMMON for Control Flags        AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
C
      IF (IO.LE.0) IO = 6       ! Standard Output Device
      WRITE(IO,10)
      WRITE(IO,20) DIMSOR,SUE
      WRITE(IO,25) BCE,2.*CMFP/DIAM
      IF (IARG.GE.2) THEN
        WRITE(IO,30) T,VPAT,CS
        WRITE(IO,32) MW,DENSTY,SURTEN
        WRITE(IO,34) DIFFUS,4.*VELQ,BCE
        WRITE(IO,35) 1.E4*AMFP,1.E4*CMFP,3.E4*BCE*CMFP
        WRITE(IO,36) 1.E4*DIAM,1.E4*DIKELV,1.E4*DSMIN
        WRITE(IO,38) RMS
      END IF
      IF (IARG.GE.1) THEN
        WRITE(IO,40) CS
        WRITE(IO,42) CS*WEIGHT
        WRITE(IO,44) 1.E8*CS*SAM        ! square microns / cc
        WRITE(IO,46) TS,TB
      END IF
      IF (IARG.GE.3) THEN
        WRITE(IO,50) DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      END IF
      IF (IARG.GE.4) THEN
        WRITE(IO,60) DOINIT,DOSORC,DODEPO,DOCOAG,NOEVAP
        WRITE(IO,62) DOCOND,DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON
        WRITE(IO,64) NUFLAG,TCON,RATEG
      END IF
      RETURN
10    FORMAT(/15X,'***  CONDENSING SYSTEM PROPERTIES  ***'/)
20    FORMAT(' Dimensionless Source Rate =',1PE10.3,4X,
     1  'Dimensionless Surface Energy =',0PF7.3 /)
25    FORMAT(' Dimensionless Diffusivity =',F7.3,5X,
     1  'Dimensionless Knudsen Number =',F8.2 /)
30    FORMAT(' T=',F5.0,' K',4X,'v.p.=',1PE10.3,' atm',4X,
     1  'Cs=',1PE10.3,' #/cc/sec')
32    FORMAT(' MW=',F7.2,4X,'Density=',F6.3,4X,'Surface Tension=',
     1  F8.3,' dyne/cm')
 34     FORMAT(' Diffusivity=',F7.4,' cm*cm/sec',4X,' VEL1=',1PE10.3,
     1  ' cm/sec',4X,'BCE=',0PF7.3)
 35     FORMAT(' Mean Free Paths:   Air=',F7.4,' um',4X,
     1  'Monomer=',F7.4,' um',4X,'Mod. F-S=',F7.4,' um')
 36     FORMAT(' Diameters:   Monomer=',F7.4,' um',4X,
     1  'Kelvin=',F7.4,' um',4X,'Minimum=',F7.4,' um')
 38     FORMAT(' Mass Source Rate=',1PE10.3,' ug/cu.m./sec'/)
 40     FORMAT(' Characteristic Number Concentration =',1PE10.3,
     1  ' #/cc  (saturated vapor)')
 42     FORMAT(' Characteristic Mass Concentration =',1PE10.3,
     1  ' ug/cu.m.  (sat.)')
 44     FORMAT(' Characteristic Surface Area =',1PE10.3,
     1  ' um**2/cm**3  (sat.)')
 46     FORMAT(' Characteristic Times (seconds):  Source =',1PE10.3,5X,
     1  'Collision = ',1PE10.3/)
 50     FORMAT(' Flags:  KELV=',L1,'  NUCL=',L1,'  CLBL=',L1,'  SCAV=',
     1  L1,'  LESSDI=',L1,'  USEBCE=',L1 /)
 60     FORMAT(' Flags:  INIT=',L1,'  SORC=',L1,'  DEPO=',L1,
     1  '  COAG=',L1,'  NOEVAP=',L1)
 62     FORMAT('           COND=',L1,'  CON2=',L1,'  LIMT=',L1,
     1  '  DVAP=',L1,'  GEOSEC=',L1,'  NCON=',L1)
 64     FORMAT('           NUFLAG=',I3,4X,'TCON=',1PE9.2,4X,'RATEG=',E9.2/)
        END
C
C----------------------------------------------------------------------
C
      FUNCTION OLDDEP(X,DUMMY,TGAS,PGAS,NBTYPE)
C
C**********************************************************************
C
C  PURPOSE:
C       To Calculate the Surface Deposition Coefficients,
C        for the processes of gravity (settling),
C        diffusion (boundary layer), and thermophoresis.
C
C  ON ENTRY:
C       X                  Log Particle Mass [ln(kg)]
C       DUMMY              Not Used
C       TGAS               Gas Temperature [K]
C       PGAS               Gas Total Pressure [Pa]
C       NBTYPE             Flag for Type of Section Coefficient:
C                          7=Ceiling   8=Vertical Walls   9=Floor
C       /GAS/   DENAIR     Background Gas Density [kg/cu.m]
C       //      FREEMP     Background Gas Mean Free Path [m]
C       //      VISCOS     Background Gas Viscosity [kg/m/sec]
C
C  ON RETURN:
C       OLDDEP             Deposition Coefficient
C
C  COMMENTS:
C       This is the original approach used in MAEROS I.
C       Based on Boundary Layer Theory
```

```
C
C     REFERENCES:  GIESEKE,J.A., LEE,K.W. AND REED,L.D. 'HAARM-3
C     USERS MANUAL,' BMI-NUREG-1991 (1978), BROCK, J.R. 'ON THE
C     THEORY OF THERMAL FORCES ACTING ON AEROSOL PARTICLES,' J.
C     COLLOID INTERFACE SCIENCE, VOL.17, 768 (1962)
C
C*********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
      V=EXP(X)                      ! Particle Mass Given
      D=ZERO                        ! Need Diameter
      CALL RHODD(V,D,RHO)           ! Calculate Diameter from Mass
C
C***             AIR VISCOSITY, DENSITY, MEAN FREE PATH HELD IN /GAS/
C***              DOUBLECHECK TEMPERATURE & PRESSURE ARE CONSISTENT
C
      IF (TGAS.NE.TEMP.OR.PGAS.NE.PRES) THEN
        IF (TGAS.NE.TEMP) TYPE 21, TEMP,TGAS
21      FORMAT(/' WARNING: /GAS/ TEMP =',F7.1,' while TGAS=',F7.1 /)
        IF (PGAS.NE.PRES) TYPE 22, PRES,PGAS
22      FORMAT(/' WARNING: /GAS/ PRES =',1PE9.2,' while PGAS=',E9.2 /)
        CALL SETGAS(TGAS,PGAS)
      END IF
C
      FCHI=CHI
      AKN=2.*FREEMP/D              ! Knudsen Number for particle in air
      BMOBIL=1.+AKN*(FSLIP+.4*EXP(-1.1/AKN))
      VTERM=.544*RHO*D*D*BMOBIL/VISCOS         ! Terminal Velocity
      DIF=1.46E-24*TGAS*BMOBIL/(VISCOS*FCHI*D)
C
      IF (NBTYPE.EQ.7) THEN
        TGRAD=TGRADC
      ELSE IF (NBTYPE.EQ.8) THEN
        TGRAD=TGRADW
      ELSE IF (NBTYPE.EQ.9) THEN
        TGRAD=TGRADF
      END IF
      VTHRML=1.5*VISCOS*BMOBIL*(FTHERM*AKN+TKGOP)*TGRAD/(FCHI*DENAIR*
     $  TGAS*(1.+3.*FSLIP*AKN)*(1.+2.*(FTHERM*AKN+TKGOP)))
      DIF=DIF/DELDEP
D      TYPE 987,DIF,D,NBTYPE
D 987  FORMAT(' DIF=',1PE10.2,3X,'FOR D=',6PF7.3,' uM',5X,'TYPE',I2)
```

```
      IF (NBTYPE.EQ.7) THEN
        OLDDEP=ACELOV*AMAX1(0.,DIF-VTERM+VTHRML)
      ELSE IF (NBTYPE.EQ.8) THEN
        OLDDEP=AWALOV*AMAX1(0.,DIF+VTHRML)
      ELSE IF (NBTYPE.EQ.9) THEN
        OLDDEP=AFLROV*AMAX1(0.,DIF+VTHRML+VTERM)
      END IF
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PEDERV(N,T,Y,PD,NO)
C
C***********************************************************************
C
C  PURPOSE:
C        To Calculate Jacobian of dQ/dt Array.  DUMMY Version!
C
C  ON ENTRY:
C        N                    Number of elements in DY/DT array
C        T                    Time [sec]
C        Y                    Dependent Array (Q in this application)
C        NO                   Actual Dimensioning of PD and Y
C
C  ON RETURN:
C        PD                   d (dQ/dt) / dQ  Matrix in One-Dim Array
C
C  COMMENTS:
C This is presently intended to be a dummy subroutine in this application
C Used only in EPISODE versions of MAEROS ; not adequate if MF=11 or 21
C If this PEDERV is actually called, program will halt.
C
C***********************************************************************
C
      TYPE 10, T
   10 FORMAT(/5X,' Error -- PEDERV was called at time ',1PE10.2/)
      TYPE 20
   20 FORMAT(' Hence MITER of MF was set equal to one'/)
      STOP 'STOP on bad MF to DRIVES for Dummy PEDERV'
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PREPLT(FNAME)
C
C***********************************************************************
C
C  PURPOSE:
C        To print a header for the PSAVE file (.DIM) giving summary
C         of some characteristic system condensation parameters.
C
C  ON ENTRY:
```

```
C       FNAME               FileName of .DIM extension (normally)
C       /NUCLO/, /NUCL1/, /NUCL2/ fixed parameters set.
C
C  ON RETURN:
C       All unchanged.
C
C  COMMENTS:
C       None.
C
C***********************************************************************
C
      CHARACTER*16 FNAME
      REAL MW
      COMMON /NUCLO/ T,VP,MW,DENSTY,SURTEN,RMS,PGAS
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT
      COMMON /NUCL2/ VL,VM,DIAM,SAM,CS,VELO,VPAT,DSMIN
      WRITE(98,91) FNAME         ! Eight Letter Title
      WRITE(98,92)
      WRITE(98,93) RMS,DIMSOR
      WRITE(98,94) TB,TS
      WRITE(98,95) VP,CS
      WRITE(98,96) DIAM,SAM,MW
   91 FORMAT(10X,'Sectional Model Simulation:  ',A16)
   92 FORMAT(' Uses Classical Nucleation, Standard Test Case')
   93 FORMAT(1P2E11.3,' Source Rate (ug/cu.m/sec and dimensionless)')
   94 FORMAT(1P2E11.3,' Time Scales (Collision & Source, in seconds)')
   95 FORMAT(1P2E11.3,' Saturation Pressure (dynes/cm*cm) & '
     $ 'Concentration (#/cc)')
   96 FORMAT(1P3E11.3,' MONOMER Diameter (cm), Surface, MW')
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PRESET(TEMP,PRES,RATEG)
C
C***********************************************************************
C
C  PURPOSE:
C       To initialize cgs /NUCLO/ from MKS PHYSPT COMMONs.
C       Used to interface standard cgs nucleation routine J with
C        the Multicomponent Aerosol Code.
C
C  ON ENTRY:
C       TEMP                Temperature [K]
C       PRES                Pressure, total [Pa]
C       RATEG               Condensible Generation Rate [kg/cu.m/sec]
C       /CONDNS/ variables set
C       /STOKES/ DENSTY (a.k.a. DENMKS) set
C
C  ON RETURN:
C       /NUCLO/ variables all set.
C       /TRANS/ DIFFUS,BCE set.
```

```
C
C  COMMENTS:
C       Should be called once at beginning by Main program.
C
C***********************************************************************
C
C***             For CGS Nucleation Subroutine:
C
      REAL MW
      COMMON /NUCLO/ T,VP,MW,DENSTY,SURTEN,RMS,PGAS
      COMMON /TRANS/ DIFFUS,DIMDIM,BCE
C
C***             From MKS Main Program:          (DENSTY & SURTEN renamed)
C
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SIGMA,DIFF,BETACE
      COMMON /STOKES/ DENMKS
      COMMON /GAS/ TEM,PRE,PSAT
C
C***             Equate or Interconvert Variables
C
      T=TEMP                     ! K from K
      VP=10.*PSAT                ! dynes/sq.cm from Pascals vapor pressure
      MW=CONMW                   ! Molecular Weight
      DENSTY=1.E-3*DENMKS        ! g/cc from kg/cu.m
      SURTEN=1.E3*SIGMA          ! dynes/cm from newtons/m
      RMS=1.E9*RATEG             ! ug/cu.m from kg/cu.m source rate
      PGAS=10.*PRES              ! dynes/sq.cm from Pascals total pressure
C
      DIFFUS=1.E4*DIFF           ! cm*cm/sec from m*m/sec
      BCE=BETACE                 ! beta in Chapmann-Enskog collision theory
C
      RETURN
      END
C
C----------------------------------------------------------------------
C
      SUBROUTINE TRSET           ! Set Transport Properties DIFFUS and BCE
C
C***             Makes Sure MKS COMMON is same as cgs COMMON for
C**              two Transport Properties
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SIGMA,DIFF,BETACE
      COMMON /TRANS/ DIFFUS,DIMDIM,BCE
      IF (DIFFUS.EQ.0.) STOP 'ERROR -- DIFFUS NOT KNOWN'
      IF (BCE.EQ.0.) STOP 'ERROR -- BCE NOT KNOWN'
      DIFF=1.E-4*DIFFUS          ! m*m/sec from cm*cm/sec
      BETACE=BCE                 ! beta in Chapmann-Enskog collision theory
      RETURN
      END
C
C----------------------------------------------------------------------
C
      SUBROUTINE PRINFO(IP,METHOD)
C
```

```
C***********************************************************************
C
C  PURPOSE:
C       To Print a Brief Header Naming the Time Integration Package and
C         Parameters Used In the Simulation.
C
C  ON ENTRY:
C       IP                  Logical Unit Number for Output Device or File
C       METHOD              CHAR*8 Name of Time Integration Package
C
C  ON RETURN:
C       All variables unchanged.
C
C  COMMENTS:
C       Nonessential subroutine; may be called once early by Main Program.
C
C***********************************************************************
C
      COMMON /PARINT/ RELE,ABSE,KTOL,MFEPI,H0    ! Integration Parameters
C        PHYSPT.INC to establish uniform COMMON for physical properties
C        COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
C        COMMON for Control Flags         AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      CHARACTER*8 METHOD
      WRITE(IP,100) '  ',METHOD,'RUN INFO  '
  100 FORMAT(' *****************************',A2,A8,A10,
     $         '*****************************'/)
      IF(DELDEP.GT.0.)  WRITE(IP,111) DELDEP
  111 FORMAT(' DELDEP IS',1PE12.4,' METERS BOUNDARY LAYER THICKNESS')
      IF (DELDEP.EQ.-1.) WRITE(IP,112) AKE
  112 FORMAT(' USING JIM CRUMPS DEPOSITION MODEL, KE=',1PE8.2)
      WRITE(IP,130) VOLUME,ACELOV+AWALOV+AFLROV
  130 FORMAT(/' CHAMBER =',F8.2,' CUBIC METERS, WITH'
     $ ' AREA:VOLUME RATIO OF',F8.4,' /M')
      WRITE(IP,135) MFEPI,RELE,ABSE,KTOL
  135 FORMAT(/' USING MF=',I3,5X,'RELE=',1PE10.3,5X,
     $ 'ABSE=',E10.3,5X,'KTOL=',I2 /)
      WRITE(IP,100) '**','********','**********'
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PRINTO(Q,TIME,VOLU,IFLAG,IPRNT)
C
C***********************************************************************
C
C  PURPOSE:
C       This routine prints outs the size distribution
C        after each specified time is reached.
C
C  ON ENTRY:
C       Q               Array of Sectional Mass Concentrations [kg/cu.m]
C       TIME            Current Time [sec]
C       VOLU            Volume of Container [cu.m]
C       IFLAG           Initialization Flag (1 if first call)
C       IPRNT           Logical Unit Number of Output Device or File
C               Also numerous COMMON block variables must be set.
C
C  ON RETURN:
C       All variables unchanged.
C
C  COMMENTS:
C       Set for 80 column wide output.
C
C***********************************************************************
C
      PARAMETER ( NEMAX = 218 )         ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)         ! Sectional Diam & Masses
      COMMON /DEPSIT/ DEPSIT(3,2)        ! Deposited Masses
C  DEPSIT array is 3 surfaces by KCOMP components.  Approximate values.
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
C       COMMON for Control Flags        AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      COMMON /CONDNS/ DELSAT,CONMW,GASMW
      COMMON /STOKES/ DENSTY
      COMMON /WALLS/ DELDEP
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT
      COMMON /NUCL2/ VL,VM,DIAM1,SAM,CS,VELQ,VPAT,DSMIN
      COMMON /INDEX/ MS,KC,NOV,NON                ! Sectional Pointers
      DIMENSION Q(NEMAX)
      DIMENSION QT(MMAX),QTV(MMAX),CUMDEP(8),QTN(MMAX)
C
      DATA DTO / 1. /    ! Initial dimensionless time (assumes saturated)
C
      QVAP=Q(NOV)                  ! Vapor Mass Concentration
      QREF=WEIGHT*CS               ! Mass Density of Saturated Vapor
      DIN=DS(1)                    ! Boundary between nucleation and condensation
C
      IF (IFLAG.EQ.1) THEN         ! IFLAG=1 to Initialize
        CUMTOT=ZERO
        DO I=1,KC
          CUMDEP(I)=ZERO           ! Initialize to no previous deposition
        END DO
      END IF
C
      SUM=ZERO
      COUNT=ZERO
      SURFAC=ZERO
      DO I=1,MS
        QT(I)=ZERO
        DO J=1,KC
          QT(I)=QT(I)+Q(J+KC*(I-1))*1.E9        ! ug/cu.m. size I
        END DO
        SUM=SUM+QT(I)              ! Note QT(I) units: ug/cu.m. total
        VHMEAN=ALOG(VS(I+1)/VS(I))/(1./VS(I)-1./VS(I+1))   ! kg mean particle
C  Remember: VS, VHMEAN is particle mass in Kilograms
        DHMEAN=ALOG(DS(I+1)/DS(I))/(1./DS(I)-1./DS(I+1))
C  Note: DS, DHMEAN is particle diameter in Meters
        FACTAV=6./DENSTY/DHMEAN                 ! sq.m. / kg aerosol
        QTN(I)=QT(I)/VHMEAN*1.E-15              ! #/CC
        COUNT=COUNT+QTN(I)
        SURFAC=SURFAC+QT(I)*FACTAV*1.E-11               ! cm*cm/cc
        QTV(I)=QT(I)*VOLU                               ! ug in size section I
      END DO
      SVOL=SUM*VOLU                             ! total ug
C
      WRITE(IPRNT,10) TIME,SUM,SVOL,(DS(I),DS(I+1),QT(I),
     $ QTV(I),QTN(I),I=1,MS)
10    FORMAT(///25X,' TIME =',1PG10.4,' SEC'//
     $ 3X,'TOTAL SUSPENDED MASS =',1PE11.4,' UG/M**3',4X,G11.4,' UG'//
     $ 8X,'DIAMETER RANGE (MICRON)',2X,'UG/M**3',8X,'UG',8X,'#/CC'/
     $ (4X,6PF10.4,' --',6PF10.4,1PE13.3,G13.3,E13.3))
      WRITE(IPRNT,11) COUNT,SURFAC
11    FORMAT(/' TOTAL NUMBER =',1PE11.3,' #/CC',6X,
     $ 'TOTAL SURFACE AREA=',E11.3,' Sq.Cm./CC')
C
```

```
      SR=SRATIO(QVAP)    ! Calculate SR from QVAP=SR*PSAT*CONMW/RGAS/TEMP
      IF (TS.GE.ZERO) THEN
        DIMT=DTO+TIME/TS
      ELSE
        DIMT=DTO+TIME/TB
      END IF
      IF (SAVDIS) WRITE(26,60) TIME,SR,DIMT
 60   FORMAT(1X,1P3E15.5,4X,'t , S , td')
      DO I=1,MS
        IF (SAVDIS) THEN
          DIMEAN=1.E6*SQRT(DS(I))*SQRT(DS(I+1)) ! mean dp in microns
          DIMQ=QT(I)/QREF
          DELX=ALOG10(DS(I+1)/DS(I))
          WRITE(26,61) DIMEAN,DIMQ,DELX
 61       FORMAT(1X,1P3E15.5)
        END IF
      END DO
      IF (KC.GT.1) THEN                    ! Not Single Component
        WRITE(IPRNT,1) (I,I=1,KC)
    1 FORMAT(/37X,'COMPONENT (UG/M**3)'/5X,'DIAMETER RANGE (MI)',
     $  8(11X,I1,1X))
C-   $  11X,'1',12X,'2',12X,'3',12X,'4',12X,'5',12X,'6',12X,'7',12X,'8')
        DO I=1,MS
          WRITE(IPRNT,19) DS(I),DS(I+1),(1.E9*Q(J+KC*(I-1)),J=1,KC)
   19     FORMAT(6PF11.4,' --',6PF10.4,2X,1P8E13.3)
        END DO
C
        DO I=1,KC
          QT(I)=ZERO
          DO L=1,MS
            QT(I)=QT(I)+Q(I+KC*(L-1))*1.E9      ! ug/cu.m. of comp I
          END DO
        END DO
        WRITE(IPRNT,34)(QT(I),I=1,KC)
   34 FORMAT(/35X,'TOTAL OF EACH COMPONENT (UG/M**3)'/26X,1P8E13.4)
        DO K=1,KC
          QT(K)=QT(K)*VOLU                       ! ug of component K
        END DO
        WRITE(IPRNT,15) (QT(K),K=1,KC)
   15   FORMAT(61X,'uG'/26X,1P8G13.3)
C
      END IF
      IF (DODVAP) WRITE(IPRNT,36) QVAP*1.E9,SR
   36 FORMAT(/1X,1PG10.4,' UG/CU.M.',5X,'SATURATION RATIO=',G13.4/)
      IF (IFLAG.EQ.1) RETURN            ! First printout so no changes
C
C***             Handle Deposition
C
      IF (DODEPO) THEN
        DO K=1,KC
          QT(K)=ZERO
          DO J=1,3
            QT(K)=QT(K)+DEPSIT(J,K)*1.E9        ! ug of component K deposited
          END DO
        END DO
        TOTDEP=ZERO
        DO K=1,KC
          TOTDEP=TOTDEP+QT(K)             ! ug total deposited in time period
        END DO
        CUMTOT=CUMTOT+TOTDEP              ! ug deposited from start time
        WRITE(IPRNT,2) TOTDEP,CUMTOT
 2      FORMAT(/15X,'TOTAL DEPOSITED MASS =',1PG10.4,' UG',3X,
     $  'CUMULATIVE =',G10.4,' UG')
C
        IF (KC.GT.1.AND.TOTDEP.GT.0.) THEN      ! Multicomponent Mass Deposite

          IF (DELDEP.GT.0.) THEN                ! Unified deposition rate
            WRITE(IPRNT,8)(1.E9*DEPSIT(1,K),K=1,KC)
 8     FORMAT(45X,'COMPONENT (uG)'/6X,'CEILING',12X,1P8G13.4)
            WRITE(IPRNT,9)(1.E9*DEPSIT(2,K),K=1,KC)
 9     FORMAT(6X,'VERTICAL WALLS',5X,1P8G13.4)
            WRITE(IPRNT,39)(1.E9*DEPSIT(3,K),K=1,KC)
 39    FORMAT(6X,'FLOOR',14X,1P8G13.4)
            WRITE(IPRNT,4)(QT(K),K=1,KC)
 4     FORMAT(/30X,'TOTAL DEPOSITED OF EACH COMPONENT (UG)'/25X,
     $        1P8G13.4)
          END IF
C
          DO K=1,KC
            CUMDEP(K)=CUMDEP(K)+QT(K)   ! Component deposition since start
          END DO
          WRITE(IPRNT,7) (CUMDEP(K),K=1,KC)
 7        FORMAT(30X,'CUMULATIVE DEPOSITED (UG)'/25X,1P8G13.4)
        END IF
      END IF
C
C***             Handle Nucleation
C
      IF (DONUCL) THEN
C         TNUC=Q(NQN)/(PI*DENSTY*(DIN**3)/6.)     ! #/cu.m. nuclei formed
C       Unfortunately DIN is inconsistent way of estimating nuclei size
         VHMEAN=ALOG(VS(2)/VS(1))/(1./VS(1)-1./VS(2))  ! kg mean particle
         TNUC=Q(NQN)/VHMEAN*1.E-6                       ! #/CC
        IF (TNUC.NE.0.) WRITE(IPRNT,190) Q(NQN)*1.E9,TNUC
 190  FORMAT(/T5,'Total Nucleation has been',1PE12.3,' ug/cu.m. or',
     $ E14.3,' #/cc')
      END IF
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PRSTAT(IPRNT)
C
C***********************************************************************
```

```
C
C  PURPOSE:
C       To Show the Number of Steps and Function Evaluations (Effort)
C        Required by the Time Integration Package (EPISODE only)
C
C  ON ENTRY:
C       IPRNT            Logical Unit Number of Output File or Device
C
C  ON RETURN:
C       All variables unchanged.
C
C  COMMENTS:
C       Useful for comparing efficiency of alternate integration
C        techniques.  Otherwise unnecessary.
C
C***********************************************************************
C
      COMMON /EPCOM9/ HUSED,NQUSED,NSTEP,NFE,NJE
      COMMON /EPCO99/ NCSTEP,NCFE,NCJE  ! Cumulative
      COMMON /EPCOMY/ YMIN
      DATA NCSTEP,NCFE,NCJE / 3*0 /      ! Initialize here
C
C***             FIND OVERALL NUMBERS OF OPERATIONS HERE
C
      MSTEP=NSTEP+NCSTEP         ! Total for whole time span
      MFE=NFE+NCFE
      MJE=NJE+NCJE
      WRITE(IPRNT,90) NSTEP,MSTEP,NFE,MFE,NJE,MJE
 90   FORMAT(/' INTEGRATION REQUIRED',2I9,3X,6H STEPS/
     1          21X,2I9,3X,14H F EVALUATIONS/
     2          21X,2I9,3X,14H J EVALUATIONS/)
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PUTCOF(ITP)
C
C***********************************************************************
C
C  PURPOSE:
C       Store COEFAV coefficients in appropriate /DBLK/ array,
C        anticipating Temperature and Pressure interpolation.
C
C  ON ENTRY:
C       ITP      Index specifying (T,P) set that COEFAV represents:
C                1=T1P1  2=T1P2  3=T2P1  4=T2P2
C       /AVGCOF/ COEFAV array holds sectional coefficients.
C
C  ON RETURN:
C       /DBLK/ (selected) array holds sectional coefficients.
C
C  COMMENTS:
```

```
C       The new package has not been tested with T,P interpolations.
C       Note the size of the CTP4 array will have to be extended,
C         if the geometric constraint is violated,
C         and the sequencing of calls to subroutines changed if
C         T and P will not be fixed.
C
C***********************************************************************
C
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      COMMON /DBLK/ CTP4(NCMAX)
C      COMMON /DBLK/ CT1P1(880),CT1P2(880),CT2P1(880),CT2P2(880)
C       NUMCOF should be no more than 880 unless ITP=1
      COMMON /INDEX/  MS,KC,NQV,NQN,
     $  NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF     ! Pointers
      COMMON /AVGCOF/ COEFAV(NCMAX)      ! Sectional Coefficients
C
      IF (ITP.GT.4 .OR. ITP.LE.0) STOP 'PUTCOF ARG ERROR'
      IBASE=(ITP-1)*NUMCOF
      IF (IBASE+NUMCOF.GT.NCMAX) STOP 'PUTCOF ERROR - TOO MANY SECTIONS'
C
C***             TRANSFER SECTIONAL COEFFICIENTS
C
      DO I=1,NUMCOF
        CTP4(IBASE+I)=COEFAV(I)
      END DO
      RETURN
      END
C
C----------------------------------------------------------------------
C
      SUBROUTINE RHODD(V,D,RHO)
C
C***********************************************************************
C
C  PURPOSE:
C       To Interconvert Particle Mass and Diameter.
C       Whichever one is set to zero will be calculated from the other.
C
C  ON ENTRY:
C       V        Particle Mass [kg]      Note: Set to 0. if to be found from D
C       D        Particle Diameter [m]   Note: Set to 0. if to be found from V
C
C  ON RETURN:
C       V, D are set.
C       RHO      (Constant) Particle Density [kg/cu.m]
C
C  COMMENTS:
```

```
C       This routine is not adequate for multicomponent aerosols with
C         components of differing densities.  As written, RHODD merely
C         returns the set DENSITY (now 1.E3 Kg/cu.m.) and interconverts
C         particle mass (V) and diameter(D).  To be more complete, a volume
C         average density over all sectional components could be used.
C
C***********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      RHO=DENSTY
      IF (V.LE.ZERO) THEN
        IF (D.GT.ZERO) THEN
          V = 0.5235987757 * D*D*D * RHO          ! Volume of Sphere
        ELSE
          TYPE 10, V,D                            ! Nothing Known
        END IF
      ELSE
        IF (D.LE.ZERO) THEN
          D = (6.*V/(PI*RHO)) ** 0.333333333      ! Diameter of Sphere
        ELSE
          TYPE 10, V,D                            ! Nothing Unknown
        END IF
      END IF
      RETURN
10    FORMAT(' RHODD Arg Error:',4X,'V=',1PE12.3,4X,'D=',1PE12.3)
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE SAVEP(TIME,Q)
C
C***********************************************************************
C
C  PURPOSE:
C       To Save Current Dimensionless Aerosol Parameters (For plotting later.)
C
C  ON ENTRY:
C       TIME            Current Time [sec]
C       Q               Sectional Mass Array [kg/cu.m]
C       /SIZES/ DS      Array of Sectional Diameters [m]
C       //      VS      Array of Sectional Masses [kg]
C
C  ON RETURN:
```

```
C       All variables unchanged
C
C  LOCAL VARIABLES:
C       DIMT    Dimensionless Time, scaled to source time to saturate
C       DIMA    Dimensionless Area, scaled to saturated vapor area
C       DIMN    Dimensionless Number, scaled to saturated vapor number
C       DIMQ    Dimensionless Mass, scaled to saturated vapor mass
C       DIMJ    Dimensionless Nucleation, scaled to mass source rate
C
C  COMMENTS:
C       Parameters saved relate to the balance between condensation
C        and nucleation.  SAVEP only called if SAVDIM is .TRUE.
C
C***********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
      COMMON /INDEX/ MS,KC,NQV  ! Sectional Pointers
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)         ! Sectional Diam & Masses
C       COMMON for Control Flags        AER:FLAGS.INC
C
      LOGICAL*1 DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,DOCON2
      LOGICAL*1 DOLIMT,DODVAP,GEOSEC,DONUCL,DOCLBL,DOSCAV,DOKELV
      LOGICAL*1 DONCON,NOEVAP,USEBCE,LESSDI
      LOGICAL*1 DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /CFLAGS/   DOINIT,DOSORC,DODEPO,DOCOAG,DOCOND,
     $                  DOCON2,DOLIMT,DODVAP,GEOSEC,DONCON,NOEVAP
      COMMON /NFLAGS/   DOKELV,DONUCL,DOCLBL,DOSCAV,LESSDI,USEBCE
      COMMON /SFLAGS/   DEBUGJ,SAVNUC,SAVDIM,SAVDIS
      COMMON /VFLAGS/   NUFLAG,TCON,RATEG
      DIMENSION Q(NEMAX),QT(MMAX),QTV(MMAX),QTN(MMAX)
C
      COMMON /DF2/ CONKEL,RJMKS ! kg/cu.m/sec
C
      COMMON /NUCL0/ T,VP,MW,DENSTY,SURTEN,SRATE
      COMMON /NUCL1/ SUE,RSCALE,TB,TS,DIMSOR,WEIGHT
      COMMON /NUCL2/ VL,VM,D1,SAM,CS,VELQ,VPAT,DSMIN
      COMMON /NUCL3/ SRO,GCRIT,DIMAA,BETAS,NFLAG,TN,RMNU,RMNMIN
C
C     SR=SRATIO(Q(NQV))            ! Get SR at this exact time
      SUM=ZERO                     ! Total Mass Concentration
      COUNT=ZERO                   ! Total Number Concentration
      SURFAC=ZERO                  ! Total Surface Area Concentration
      DIAMT=ZERO                   ! Total Linear Concentration
C
C***           SUM FOR TOTAL MASS, NUMBER, SURFACE AREA, DIAMETER
```

```
C
      DO I=1,MS
        QT(I)=ZERO
        DO J=1,KC
          QT(I)=QT(I)+Q(J+KC*(I-1))*1.E9        ! ug/cu.m. size I
        END DO
        SUM=SUM+QT(I)             ! Note QT(I) units: ug/cu.m. total
        VHMEAN=ALOG(VS(I+1)/VS(I))/(1./VS(I)-1./VS(I+1))  ! kg mean particle
C   Remember: VS, VHMEAN is particle mass in Kilograms
C   VHMEAN is factor relating mass and number in a section, for q(x) constant.
        DHMEAN=ALOG(DS(I+1)/DS(I))/(1./DS(I)-1./DS(I+1))
C   Note: DS, DHMEAN is particle diameter in Meters
C   DHMEAN is mean diameter relating surface and volume in a section.
        FACTAV=6./DENSTY/DHMEAN                 ! sq.m. / g aerosol **
        QTN(I)=QT(I)/VHMEAN*1.E-15              ! #/cc
        D2MEAN=2.*ALOG(DS(I+1)/DS(I))
        D2MEAN=D2MEAN/(1./DS(I)**2-1./DS(I+1)**2)
        FACTDV=6./PI/DENSTY/D2MEAN              ! tot m / g aerosol
        COUNT=COUNT+QTN(I)                      ! #/cc
        SURFAC=SURFAC+QT(I)*FACTAV*1.E-14               ! cm*cm/cc
        DIAMT=DIAMT+QT(I)*FACTDV*1.E-13         ! cm/cc
        QTV(I)=QT(I)*VOLU                       ! ug in size section I
      END DO
      SVOL=SUM*VOLU                             ! total ug in chamber
C
C***              CALCULATE TRANSIENT DIMENSIONLESS PROPERTIES
C
      PHI2=-1.                            ! Often Phi2 is indeterminant
      CORATE=1.E9*CONKEL                  ! mass rate of condensation, ug/cu.m
      DIMT=TIME/TS+1.                     ! Assumes SRO=1.
      DIMA=SURFAC/(CS*SAM)
      DIMN=COUNT/CS
      DIMQ=SUM/WEIGHT/CS
      IF (SRATE.NE.ZERO) DIMJ=RMNU/SRATE
      IF (RMNU+CORATE.NE.ZERO) PHI2=RMNU/(CORATE+RMNU)
      IF (SRATE.NE.ZERO) PHI3=(CORATE+RMNU)/SRATE
      RNJ=RMNU/WEIGHT*(D1/DSMIN)**3
      WRITE(98,98) TIME,DIMT,SRO,DIMA,DIMN,DIMQ,DIMJ,-1.,PHI2,PHI3,RNJ
   98 FORMAT(1PE11.3,0P2F11.4,1P4E11.3,0PF11.6,1P3E11.3)
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE SETGAS(TGAS,PGAS)
C
C***********************************************************************
C
C  PURPOSE:
C       To set gas properties kept in /GAS/ COMMON.
C
C  ON ENTRY:
C       TGAS            Gas Temperature [K]
```

```
C       PGAS              Gas Total Pressure [Pa]
C
C  ON RETURN:
C       /GAS/   TEMP    Gas Temperature [K]
C       //      PRES    Gas Total Pressure [Pa]
C       //      PSAT    Saturation Vapor Pressure [Pa]
C       //      DENAIR  Gas Density [kg/cu.m]
C       //      FREEMP  Gas Mean Free Path [m]
C       //      VISCOS  Gas Viscosity
C
C  COMMENTS:
C       "Gas" refers to the background gas, in this case, air.
C       This SETGAS version is for air only.
C       Routine only called once unless temperature or pressure change.
C
C***********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )   ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )      ! MKS
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
      COMMON /CONDNS/ DELSAT,CONMW,GASMW
C
      TEMP=TGAS
      PRES=PGAS
C       PSAT should be determined as a function of TEMP.
C       For now it is assumed PSAT was set earlier and is fixed.
      DENAIR=1.21E-4*PGAS*GASMW/TGAS
      VISCOS=.003661*TGAS
      VISCOS=.0066164*VISCOS*SQRT(VISCOS)/(TGAS+114.)
      FREEMP=VISCOS/DENAIR*SQRT(1.89E-4*GASMW/TGAS)
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      FUNCTION SRATIO(QVAP)      ! Finds SRATIO using MKS values
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Current Saturation Ratio.
C
C  ON ENTRY:
C       QVAP              Vapor Mass Concentration [kg/cu.m]
C       /CONDNS/CONMW     Molecular Weight of Condensible
C       /GAS/   TEMP      Vapor Temperature [K]
C       //      PSAT      Vapor Pressure of Condensible [Pa]
C
C  ON RETURN:
C       SRATIO            Saturation Ratio (P1/PSAT) [-]
C
```

```
C  COMMENTS:
C        Used when a D.E. is used to follow the vapor concentration,
C         i.e., when DODVAP is .TRUE.
C        Note QVAP = Q(NQV), where vapor subscript NQV=MS*KC+1
C
C*********************************************************************
C
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. )    ! PCONS.INC
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.3144E3 )       ! MKS
      COMMON /GAS/ TEMP,PRES,PSAT,DENAIR,FREEMP,VISCOS  ! Gas Properties
      COMMON /CONDNS/ DELSAT,CONMW        ! CONMW needed
      SRATIO=QVAP*RGAS*TEMP/CONMW/PSAT  ! MKS Partial Pressure Ratio
      RETURN
      END
C
C---------------------------------------------------------------------
C
      FUNCTION SSKELV(SR,DP,DIKELV)
C
C*********************************************************************
C
C  PURPOSE:
C        To Compute the Kelvin Effect, giving effective supersaturation.
C
C  ON ENTRY:
C        SR                 Saturation Ratio (Bulk)
C        DP                 Particle Diameter [m]
C        DIKELV             Characteristic Kelvin Diameter [m]
C                            = 4. Surten Vm / kT = Dcrit log(SR)
C
C  ON RETURN:
C        SSKELV             Effective Supersaturation at Spherical Surface
C
C  COMMENTS:
C        DP and DIKELV need only have the same units.
C        DIKELV is closely related to the critical diameter.
C
C*********************************************************************
C
      DD = DIKELV / DP                     ! Dimensionless Diameter
      SSKELV = SR - EXP ( DD )             ! Condensation - Evaporation
      RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE STORE(IODIR,NEWCOF,TGAS,PGAS,IPRNT,CNAME)
C
C*********************************************************************
C
C  PURPOSE:
C        To Store/Restore Sectional Coeffients To/From a Data File.
```

```
C        This saves the effort of recalculating coefficients each
C         time the program is run.
C
C   ON ENTRY:
C        IODIR              Determines Direction of Data Transfer:
C                             0 = Output to File    1 = Input from File
C        NEWCOF             Flag to control calculation of sectional coef.
C        TGAS               Gas Temperature [K]
C        PGAS               Gas Total Pressure [Pa]
C        IPRNT              Logical Unit Number for Output Messages
C        CNAME              Coefficient File Name (CHAR*20)
C
C   ON RETURN:
C        COEFAV array is filled if IODIR=1
C        All other variables unchanged.
C
C*********************************************************************
C
      PARAMETER ( NEMAX = 218 )          ! NEMAX.INC : 218 Simultaneous ODEs
      PARAMETER ( MKMAX=NEMAX-2 )                ! Maximum Diff. Eq. for Q's
      PARAMETER ( MMAX=108 , MMAX1=MMAX+1 )      ! Maximum Sections
      PARAMETER ( NCMAX=2*MMAX*(2+MMAX) )        ! Number Coefficients
      PARAMETER ( NWMAX=6*NEMAX+3 )              ! WORK Array
C       Now set for 36 sections by 2 components plus one vapor component
C       PHYSPT.INC to establish uniform COMMON for physical properties
C       COMMON Variables Initialized and Described in APDATA.INC
C
      COMMON /CHAMBR/ ACELOV,AFLROV,AWALOV,VOLUME
      COMMON /WALLS/  DELDEP,TURBDS,AKE
      COMMON /CONDNS/ DELSAT,CONMW,GASMW,SURTEN,DIFFUS,BCE
      COMMON /STOKES/ DENSTY,CHI,FSLIP,STICK,GAMMA
      COMMON /THERM/  FTHERM,TGRADC,TGRADF,TGRADW,TKGOP
      COMMON /INDEX/  MS,KC,NQV,NQN,
     $  NB2A,NB2B,NB3,NB4,NDEPST,NGROW,ICONDN,NUMCOF     ! Pointers
      COMMON /AVGCOF/ COEFAV(NCMAX)      ! Sectional Coefficients
      COMMON /SIZES/ DS(MMAX1),VS(MMAX1)         ! Sectional Diam & Masses
      EQUIVALENCE (PPROP1,ACELOV),(PPROP2,DELDEP),(PPROP3,DELSAT),
     $            (PPROP4,DENSTY),(PPROP5,FTHERM)
      CHARACTER*20 CNAME                 ! Coefficient File Name
      CHARACTER*6 AJ                     ! Dummy to Read in Label (of COMMON)
      DIMENSION DIAM(MMAX1)              ! Diameter consists of MS+1 elements
      DIMENSION PPROP1(4),PPROP2(3),PPROP3(6),PPROP4(5),PPROP5(5)
      DIMENSION OPROP1(4),OPROP2(3),OPROP3(6),OPROP4(5),OPROP5(5)
C       Labels: CHAMBR    WALLS     CONDNS    STOKES    THERM
C
      IF (IODIR.EQ.0) THEN      ! On IODIR=0, Output to File CNAME (.CO)
        OPEN (UNIT=2,FILE=CNAME,STATUS='NEW')
        WRITE(2,29)
        WRITE(2,30) NEWCOF,MS,KC,TGAS,PGAS
        WRITE(2,31) 'CHAMBR',PPROP1
        WRITE(2,31) 'WALLS ',PPROP2
        WRITE(2,31) 'CONDNS',PPROP3
        WRITE(2,31) 'STOKES',PPROP4
        WRITE(2,31) 'THERM ',PPROP5
        WRITE(2,32) NB2A,NB2B,NB3,NB4,NDEPST,NGROW,NUMCOF
        WRITE(2,38)
        WRITE(2,33) (DS(I),I=1,MS+1)
        WRITE(2,*) 'BETA 1B (Growth from Adjacent Sections)'
        WRITE(2,33) (COEFAV(I),I=1,NB2A)
        WRITE(2,*) 'BETA 2A (Loss by Coagulation with Smaller)'
        WRITE(2,33) (COEFAV(I),I=NB2A+1,NB2B)
        WRITE(2,*) 'BETA 2B (Gain by Coagulation with Smaller)'
        WRITE(2,33) (COEFAV(I),I=NB2B+1,NB3)
        WRITE(2,*) 'BETA 3B (Self Coagulation Losses)'
        WRITE(2,33) (COEFAV(I),I=NB3+1,NB4)
        WRITE(2,*) 'BETA 4  (Loss by Coagulation with Larger)'
        WRITE(2,33) (COEFAV(I),I=NB4+1,NDEPST)
        WRITE(2,*) 'WALL DEPOSITION (per second)'
        WRITE(2,33) (COEFAV(I),I=NDEPST+1,NGROW)
        WRITE(2,*) 'CONDENSATIONAL GROWTH'
        WRITE(2,33) (COEFAV(I),I=NGROW+1,NUMCOF)
   22 FORMAT(1X)
   29 FORMAT(' -----  MAEROS COEFFICIENT FILE  -----')
   30 FORMAT(' NEWCOF=',I3,3X,'MS=',I3,3X,'KC=',I3,3X,
     $ 'TGAS=',1PG16.8,3X,'PGAS=',1PG16.8)
   31 FORMAT(1X,A6,4X,1P7G16.8)
   32 FORMAT(' INDICES:',5X,7I6)
   33 FORMAT(1P8G16.8)
   38 FORMAT(' SECTIONAL DIAMETERS IN METERS')
   40 FORMAT(' NEWCOF=',I3,3X,'MS=',I3,3X,'KC=',I3,3X,
     $ 'TGAS=',G16.8,3X,'PGAS=',G16.8)
   41 FORMAT(1X,A6,4X,7G16.8)
   43 FORMAT(8G16.8)
C
      ELSE                ! On IODIR=1, Read coefficients from STORAGE.CO
        OPEN (UNIT=2,FILE=CNAME,STATUS='OLD')
        READ(2,22)
        READ(2,40) JNEWCO,JMS,JKC,OTGAS,OPGAS
        IF (JMS.NE.MS.OR.OTGAS.NE.TGAS.OR.OPGAS.NE.PGAS) THEN
          TYPE 30, JNEWCO,JMS,JKC,OTGAS,OPGAS
          GOTO 900
        END IF
        READ(2,41) AJ,OPROP1
        READ(2,41) AJ,OPROP2
        READ(2,41) AJ,OPROP3
        READ(2,41) AJ,OPROP4
        READ(2,41) AJ,OPROP5
        DO I=1,4
          IF (PPROP1(I).NE.OPROP1(I)) GOTO 900
        END DO
        DO I=1,3
          IF (PPROP2(I).NE.OPROP2(I)) GOTO 900
        END DO
        DO I=1,6          ! Ignore SURTEN as no COEFF effect
          IF (I.NE.4.AND.PPROP3(I).NE.OPROP3(I)) GOTO 900
        END DO
```

```
          DO I=1,5
            IF (PPROP4(I).NE.OPROP4(I)) GOTO 900
          END DO
          DO I=1,5
            IF (PPROP5(I).NE.OPROP5(I)) GOTO 900
          END DO
          READ(2,32) NB2A,NB2B,NB3,NB4,NDEPST,NGROW,NUMCOF
          IF (NUMCOF.EQ.0) NUMCOF=NGROW+3*MS-1
          READ(2,22)
          READ(2,43) (DIAM(I),I=1,MS+1)
          DO I=1,MS+1
            IF (DS(I).NE.DIAM(I)) GOTO 900
          END DO
          READ(2,22)
          READ(2,43) (COEFAV(I),I=1,NB2A)
          READ(2,22)
          READ(2,43) (COEFAV(I),I=NB2A+1,NB2B)
          READ(2,22)
          READ(2,43) (COEFAV(I),I=NB2B+1,NB3)
          READ(2,22)
          READ(2,43) (COEFAV(I),I=NB3+1,NB4)
          READ(2,22)
          READ(2,43) (COEFAV(I),I=NB4+1,NDEPST)
          READ(2,22)
          READ(2,43) (COEFAV(I),I=NDEPST+1,NGROW)
          READ(2,22)
          READ(2,43) (COEFAV(I),I=NGROW+1,NUMCOF)
        END IF
        CLOSE (2)
        RETURN
C
  900 WRITE(IPRNT,910) CNAME              ! CNAME is for different conditions
  910 FORMAT(/' ***  PROPERTIES INCONSISTENT WITH ',A20,'  ***'/)
      CLOSE (2)
      IODIR=-1             ! Flag that file was not appropriate
      RETURN               ! Program must compute it's own COEFAV
      END
C
C------------------------------------------------------------------------
C
      SUBROUTINE DRIVE (N, TO, HO, YO, TOUT, EPS, IERROR, MF, INDEX)
C
C************************************************************************
C
C  PURPOSE:
C        To Solve a System of Stiff ODEs, with custom modifications to
C         handle a non-negativity constraint and to keep error limited
C         where neither simple relative nor absolute error bounds
C         are appropriate.
C
C  ON ENTRY:
C        See original documentation below.
C
```

NOTE: To avoid duplication and save considerable space, the DRIVE subroutine and associatiated subroutines comprising the EPIS (custom-modified EPISODE) package are omitted here. They may be found in full at the end of the RSNM listing in Appendix B. (Trivial modifications were made to convert from VAX FORTRAN-77 to MICROSOFT FORTRAN v3.20 running on the IBM AT or XT systems.)

APPENDIX B:

LISTING OF REVISED SNM CODE

The following pages contain a source listing of the RSNM (Revised Saturation-Number-Mass) code. This is a lengthy implementation of the SNM aerosol nucleation and growth model, allowing the user a wide variety of options when running the program. The solution of the five simultaneous ODE's comprising the dual-mode dimensionless SNM model is achieved by the EPIS (modified Gear) integration routine. Time may be scaled to either source rate or monomer-monomer collision rate, as appropriate, and initial supersaturations, vapor sources, and wall losses may be included in or omitted from the simulations.

The RSNM simulation is run on the IBM XT or AT personal computers. The executable program is created by linking together the object files compiled from RSNM.FOR, DRSNM.FOR, DIMSET.FOR, DBPSET.FOR, RSET2.FOR, DEPSIT.FOR, PDO.FOR and EPIS.FOR, which are listed on the following pages. The user may design an RSET2 subroutine to run through a set of simulations all at once (the listing includes an RSET2 subroutine used to simulate Dr. Okuyama's DBP data with seed aerosol). If the user wishes to use automatic property evaluation (specifying only temperature) for a condensable vapor other than DBP, the SETDBP subroutine must be replaced. The wall loss expression is taken from McMurry's work to include electrostatic forces.

The RSNM program generates several ASCII files for the user to inspect or plot. They consist of the following:

Unit 11 [EPI.] Summary of EPIS time integration and problems.

Unit 50 [RSNM.] Text summary of each simulation.
SRO, DIMNO, DIMN, DIMT
(SRO, CINIT, TEMP, TOTN, TIME)

Unit 60 [SUM.] Nondimensional summary for plotting.
SRO, DIMST, DIMKN, DIMNO, DIMDO, DIMN, DIMD

Unit 70 [RUN.] Profile of system evolution with time.
DIMT, S, DIMN, DIMM, DIMD, DIMC
(DIMT, S, DIMN, DIMM, TIME, TOTN, DPX, DPC)

Unit 80 [SUM2.] Dimensional summary for plotting.
SRO, SIGMA, VPAT, TEMP, CINIT, DPI, TOTN, DPX

Unit 90 [SUPPRESS.] Initial aerosol suppression summary.
SRO, DPIM, CINIT, TOTN, N1/NJO, N2/NJO, N/(N1+NJO), N/NJO
(SRO, DIMDO, DIMNO, DIMN, N1/NJO, N2/NJO, N1/(N1+NJO), N/NJO)

Unit 99 [DEBUG.] User selected info (none by default).

The variables listed in parentheses are used in place of the regular variable list going to any file whenever the user asks for the output to be in dimensional form. Also note that DIMSOR replaces SRO for any simulations driven by a dimensionless source rate rather than by (just) an initial supersaturation.

```
$DEBUG
      PROGRAM RSNM

C*********************************************************************

C  Roof Saturation-Number-Mass model.
C  This is a dual-mode dimensionless S-N-M model by DRW
C  Deposition added for smog chamber data analysis by JES
C  Adapted for Microsoft FORTRAN-77 on the AT by DRW   2/12/86

C        Runs with fixed source rate for roof lab simulations.
C        Link with DRSNM (DIFFUN), DIMSET, DBPSET, DEPSIT, EPIS, and PDO.

C        User may change the physical parameters of the simulation
C         or the input parameters.  Defaults given.

C  The RSNM simulation gives the user interactive abilities to

C   - set physical properties or use PRESET values,
C          which may be a function of temperature
C   - choose between source and initial-condition driven system
C          with corresponding scaling to TS or to TB
C   - include or omit deposition, setting parameters
C   - repeat for different values (optionally PRESET) of
C          - dimensionless source rate or initial saturation ratio
C          - aerosol number
C          - aerosol mean diameter
C   - use automatic time selection
C   - set integration routine parameters


C*********************************************************************

      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. , THREE=3. )
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( RGAS = 8.31433E+7 )
      PARAMETER ( BK = 1.38054E-16 )

C  Most variables, including constants, are cgs units.

C  Allocate S,RN1,RM1,RN2,RM2 array space

      REAL X(5),DXDT(5)
      EQUIVALENCE (X(1),S)
      EQUIVALENCE (X(2),RN1) , (X(3),RM1)
      EQUIVALENCE (X(4),RN2) , (X(5),RM2)

      REAL KE, MW, MW1
      INTEGER REGIME
      CHARACTER*1 ASK

C---------------------------------------------------------------------
```

```
      LOGICAL CUT, CHECK, DEBUG, SAVSUM, SAVRUN, DOPLOT
      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      LOGICAL REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /REPEAT/ REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /OFLAGS/ NDISP,CUT,CHECK,DEBUG,SAVSUM,SAVRUN,DOPLOT
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      COMMON /XARRAY/ X
      COMMON /CASE/ NCASE,RNJO

C    *  DOKELV -- includes the Kelvin effect on condensation
C    *  USEBCE -- uses Chapmann Enskog instead of modified Fuchs-Sutugin
C    *  SCALES -- use TS (source) rather than TB (collision) time scaling
C    *  HIGHJ  -- artificially boost nucleation rate by FACTJ
C    *  LOTHE  -- use Lothe-Pound nucleation rate expression
C    *  DODEPO -- includes electrostatic deposition
C       REGIME -- 0 for normal, 1=kinetic limit, 2=continuum limit
C       FACTJ  -- if (HIGHJ), nucleation rates multiplied by this factor.

C                    COMMON for derivative routine

      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BETACE,GS,FIN1,FIN2,FACTJ

C                    COMMON for nucleation routine

      COMMON /SNMX/ REGIME,DIAM1,CS,MW1,RSCAL1,TB,TS
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     $                   DENSTY

C                    COMMON for deposition routine

      COMMON /DEPOST/ ELCFLD,KE,DI1,DVESSL,RHO,DIFF,ICHRG,TGAS

C                    COMMON for integration routine

      COMMON /EPCOMY/ YMIN,HMAX

C                    COMMON for Output of Results

      COMMON /INIT/ SR0,DIMN0,DIMD0,CINIT,DPI,TEMP,CSAT

      COMMON /RESULT/ DIMT,DIMN,DIMM,DIMA,DIMD,TIME,TOTN,TOTM,TOTA,DPX

      COMMON /TIMES/ TSCALE,TSTEP,TMAX,TRES

      COMMON /EPC099/ NCSTEP,NCFE,NCJE
      NCSTEP = 0
      NCFE = 0
      NCJE = 0
```

```
C***            Set Integration Routine Parameters  (Leave Alone)

C                    Five Simultaneous Equations
      NEQ = 5

      KTOL = 5

      MF = 20


C***        INITIALIZE DATASTREAM FLAGS (User may modify these following)

C                    First Call prints out input values
      CHECK = .TRUE.

C                    All Calls print out calculated values
      DEBUG = .FALSE.

C                    Stops Time integration if S<SMIN
      CUT = .FALSE.

C                    Saves time profiles (as goes to screen)
      SAVSUM = .TRUE.

C                    Selects Monitor display format for Simulations
      NDISP = 3

C                    Saves record of S,t*,N*,M*,G*,Dp,Dc,Np
      SAVRUN = .TRUE.

C                    Saves Plottable Record
      DOPLOT = .FALSE.

C                    Used only for Suppression runs; avoid /0.
      RNJO = 1.

C
      NODIM = .FALSE.
      NOSHO = NODIM
      CINIT = 5000.
      DPI = 0.06E-4
      DIMSOR = 1.E-4
      TEMP = 298 - 273.16

      DIMST = 10.
      DIMKN = 200.
      BCE = 1./3.

      RN1 = 0.
      RM1 = 0.

C  *                 Relative Error for integration routine
      RELERR = 1.E-4
```

```
C  *                 Absolute Error for integration routine
      ABSERR = 1.E-20
C
C  *        Maximum internal time step size for integration routine
      HMAXO = 1.

C                    Nucleated Cluster Number (>gc necessary)
      GS=500.

C  Polydispersity Factor for 1) Initial & 2) Nucleated Aerosol
      FIN1=1.0
      FIN2=1.0

C  *                 Unity Saturation Ratio
      SRO=ONE

C  *                 Any Residence Time (seconds) fixes TMAX.
      TRES = 0.

C  *                 Dimensionless Time Step for Printout
      TSTEP = 0.05

C-----------------------------------------------------------------

C           *** SUMMARY OF SAVED DATA FILES PRODUCED ***

C  11:      >EPI.   Summary of EPIS integration values & problems.

C  50:      >RSNM.  Textual Summary of Each Simulation.
C           SRO,DIMNO,DIMN,DIMT                         (NOSHO)
C           SRO,CINIT,TEMP,TOTN,TIME                    (.NOT.)

C  60:      >SUM.   Nondimensional Summary for ZPLOT.
C           SRO,DIMST,DIMKN,DIMNO,DIMDO,DIMN,DIMD

C  70:      >RUN.   Profile of System Evolution.
C           DIMT,S,DIMN,DIMM,DIMD,DIMC                  (NOSHO)
C           DIMT,S,DIMN,DIMM,TIME,TOTN,DPX,DPC          (.NOT.)

C  80:      >SUM2.  Dimensional Summary for ZPLOT.
C           SRO,SIGMA,VPAT,TEMP,CINIT,DPI,TOTN,DPX

C  90:      >SUPPRESS.      Initial Aerosol Suppression Summary.
C           SRO,DPIM,CINIT,TOTN,N1/NJO,N2/NJO,N/(N1+NJO),N/NJO
C           SRO,DIMDO,DIMNO,DIMN, . . .           (.NOT.NOSHO)

C  99:      >DEBUG.         User selected info (none by default).

C  Note: Above variables are for (.NOT.USOR); if (USOR), then
C           DIMSOR replaces SRO (except is RSMASS in SUM2.).

      OPEN (UNIT=11,FILE='EPI.',STATUS='NEW')
      OPEN (UNIT=50,FILE='RSNM.',STATUS='NEW')
      OPEN (UNIT=60,FILE='SUM.',STATUS='NEW')
      OPEN (UNIT=70,FILE='RUN.',STATUS='NEW')
      OPEN (UNIT=80,FILE='SUM2.',STATUS='NEW')
      OPEN (UNIT=90,FILE='SUPPRESS.',STATUS='NEW')
      OPEN (UNIT=99,FILE='DEBUG.',STATUS='NEW')


C----------------------------------------------------------------------

C***            Input simulation parameters, allowing defaults
C  These will not change between simulations

      CALL RASK1 (RELERR,ABSERR,HMAXO,NDISP)
      IF (.NOT.USOR) TSTEP = 5.

      YMIN = ABSERR
      HMAX = HMAXO
      NCASE = 0

C----------------------------------------------------------------------

C  Accept parameters which the user may interactively change
C   for parallel simulations in a single run of RSNM.

C  User may input:  DIMSOR, CINIT, DPI, SRO, TEMP

 100  NCASE = NCASE + 1
      IF (PRESET) THEN

C  Note PRESET supersedes almost all of RASK1 so this
C  RSET2 subroutine is responsible for setting RASK1 values too!

         CALL RSET2(NCASE,DIMSOR,TRES)
         T = TEMP + 273.16
         IF (UTEMP) THEN
            CALL DBPSET(T,DIMST,DIMKN,TB,CS)
          ELSE
            CALL DIMSET(DIMST,DIMKN,TB,TS)
         ENDIF

       ELSE IF (NODIM) THEN

         CALL RASK2D(NCASE,DIMSOR,DIMST,DIMKN)

       ELSE

         CALL RASK2(NCASE,DIMSOR,FACTJ)
         T = TEMP + 273.16

         IF (UTEMP) THEN
            CALL DBPSET(T,DIMST,DIMKN,TB,CS)
          ELSE
          CALL DIMSET(DIMST,DIMKN,TB,CS)
        ENDIF

      ENDIF

      IF (NCASE.LT.0) GOTO 900

C-----------------------------------------------------------------

C          Following Code is executed regardless of PRESET

 150    IF (NODIM) GOTO 160
        IF (USOR) TS = TB/DIMSOR
        IF (SCALES) THEN
           TSCALE = TS
         ELSE
           TSCALE = TB
        ENDIF


C          Equate DIFFUN's /SNMX/ COMMON to DIMSET's /DS/ COMMON

        DIAM1=DIAM
        MW1=MW
        RSCAL1=RSCALE
        BETACE=BCE

C          Equate /INIT/ CSAT to /DS/ CS

        CSAT = CS

C          Equate /DEPOST/ COMMON to /DS/ COMMON

        DI1=DIAM
        RHO=DENSTY
        DIFF=DIFFUS
        TGAS=T

C-----------------------------------------------------------------

C          PREPARE FOR INTEGRATION

C***            Initialize Independent Variable: Nondimensional Time

 160    TDIM=ZERO

C***            Initialize Dependent Variables  (All Nondimensional)

        S=SR0

C                 Always No Secondary Aerosol Number at t=0
        RN2=ZERO
```

```
C                  Always No Secondary Aerosol Mass at t=0
        RM2=ZERO

        IF (NODIM) THEN
           NOSHO = .TRUE.
         ELSE
           DIMNO = CINIT / CS
           DIMDO = DPI/DIAM
        ENDIF

        RN1 = DIMNO
        RM1 = RN1 * DIMDO**3

C***              Note Primary (1) is defined in terms of pre-existing aerosol
C***              and includes "secondary" aerosol which condenses onto
C***              a primary particle.  Secondary (2) aerosol is defined
C***              as homogeneously nucleated particles, including their growth.

C-----------------------------------------------------------------

C  Figure out appropriate Step Sizes for Integrator & Printout

      CALL RSTEP (HO, HMAXO, HMAX, SCALES)

C  Initial Printout

      CALL RSHOW (RELERR, NCASE)

      CALL HEADER (NDISP, UHOURS)

C-------------------------------------------------------------

C          PERFORM TIME INTEGRATION

      INDEX = 1

      DIMT = TDIM
      IF (SCALES) DIMT = DIMT + SRO
      CALL ROUT (TDIM)

      KMAX = TMAX/TSTEP
c       WRITE(99,299) TMAX,TSTEP,SRO
c299    FORMAT(' Tmax=',1pe10.2,'  Tstep=',1pe10.2,' SRO=',0PF9.3)

      IF (KMAX.LE.10) KMAX=10

      DO 200 KS = 1, KMAX

         PS=S
         PN=DIMN
         PT=TDIM
         TOUT = FLOAT(KS) * TSTEP
      CALL DRIVE(NEQ,TDIM,HO,X,TOUT,RELERR,KTOL,MF,INDEX)

      IF (TDIM.EQ.0.) THEN
         WRITE(*,*) 'DRIVE failed to advance in time.'
         GOTO 500
      ENDIF

      DIMT = TDIM
      IF (SCALES) DIMT = DIMT + SRO
      CALL ROUT (TDIM)

      IF (DIMN.EQ.PN.AND.S.LT.PS.AND.DIMN.NE.0. .OR.
   1          CUT.AND.S.LT.SMIN .OR. DIMN.LT.1.E-15*DIMNO) THEN
         GOTO 500
      ENDIF

      IF (KS.EQ.20*(KS/20)) CALL HEADER (NDISP, UHOURS)

 200  CONTINUE

C  Ends Loop thru Output Times

C-------------------------------------------------------------

C          Do Next Simulation

 500  CALL RSUM (S)
      IF (REPEAT) GOTO 100

C          Program is finished

 900  STOP 'RSNM reached normal completion.'
      END


C=============================================================

      SUBROUTINE RASK1 (RELERR,ABSERR,HMAXO,NDISP)

C  This Subroutine Interactively allows the user to change
C   a variety of parameters for the RSNM simulations.
C  These parameters are then assumed fixed for the duration
C   of the exectution of RSNM.

C                    PRESET, DODEPO, UTEMP, HIGHJ, FACTJ
C                    SIGMA, VP, ELCFLD, ICHRG, KE, DVESSL
C  Candidates:       USEBCE, LOTHE

      REAL KE, MW, MW1
      INTEGER REGIME
      CHARACTER*1 ASK
      LOGICAL TRUTH
      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      LOGICAL REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /REPEAT/ REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO

C    *  DOKELV -- includes the Kelvin effect on condensation
C    *  USEBCE -- uses Chapmann Enskog instead of modified Fuchs-Sutugin
C    *  SCALES -- use TS (source) rather than TB (collision) time scaling
C    *  HIGHJ  -- artificially boost nucleation rate by FACTJ
C    *  LOTHE  -- use Lothe-Pound nucleation rate expression
C    *  DODEPO -- includes electrostatic deposition
C       REGIME -- 0 for normal, 1=kinetic limit, 2=continuum limit
C       FACTJ  -- if (HIGHJ), nucleation rates multiplied by this factor.

C                     COMMON for derivative routine

      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BETACE,GS,FIN1,FIN2,FACTJ

C                     COMMON for nucleation routine

      COMMON /SNMX/ REGIME,DIAM1,CS,MW1,RSCAL1,TB,TS
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     $                DENSTY

C                     COMMON for deposition routine

      COMMON /DEPOST/ ELCFLD,KE,DI1,DVESSL,RHO,DIFF,ICHRG,TGAS

C                     COMMON for integration routine

      COMMON /EPCOMY/ YMIN,HMAX


C-----------------------------------------------------------------

C  Set COMMON Variables - These defaults are normally fixed!

C                     Normally Transport Regime=0 (unless testing)
      REGIME = 0

C                               Include Kelvin Effect
      DOKELV = .TRUE.

C                     MFS (so BCE=1/3.) or Chapmann-Enskog
      USEBCE = .FALSE.

C--------------------------------------------------------------------

C  *                  True means a vapor source is present
      USOR = .TRUE.

C  *                  True will Scale to Source Rate TS not TB
```

```
      SCALES = USOR

C  *                 Include Deposition
      DODEPO = .FALSE.

C                    Turn off multiple simulation flags
      NEWSR = .FALSE.
      NEWAER = .FALSE.
      NEWSIZ = .FALSE.
      NEWTEM = .FALSE.

C  *                 No enhanced nucleation rate
      HIGHJ  = .FALSE.

C                    Artificially boost nucleation rate if (HIGHJ)
      FACTJ = 1.0E+8

C                    No Lothe-Pound nucleation (use Classical BDZ)
      LOTHE  = .FALSE.

C             Default simulation parameters:

C  These should very roughly approximate the roof lab experiments.

      SIGMA = 25.
      VP = 1.E-5
      ELCFLD = 40.
      ICHRG = 2
      KE = 0.01
      DVESSL = 100.

C-----------------------------------------------------------------------

      WRITE(*,234)
 234  FORMAT(' Enter Transport Regime (C/K/G) [General] : ',\)
      READ(*,11) ASK
      IF (ASK.EQ.'K' .OR. ASK.EQ.'k') REGIME=1
      IF (ASK.EQ.'C' .OR. ASK.EQ.'c') REGIME=2

      WRITE(*,9)
    9 FORMAT(/' Shall we use the PRESET set of simulations [N] ? ',\)
      READ(*,11) ASK
   11 FORMAT(A1)
      PRESET=TRUTH(ASK,'N')
      IF (PRESET) THEN
         REPEAT = .TRUE.
         GOTO 90
      ENDIF

      WRITE(*,70)
 70   FORMAT('/Will there be a vapor source term [Y] ? ',\)
      READ(*,11) ASK
      USOR = TRUTH(ASK,'Y')
      SCALES = USOR
      IF (.NOT.USOR) DIMSOR=0.

      WRITE(*,22)
   22 FORMAT(/' Will only dimensionless properties be used [N] ? ',\)
      READ(*,11) ASK
      NODIM=TRUTH(ASK,'N')
      NOSHO=NODIM

      IF (NODIM) GOTO 90

      IF (.NOT.PRESET) THEN
         WRITE(*,14)  'N'
   14    FORMAT(/' Will properties be calculated from temperature ',
     $         '[',A1,'] ? ',\)
         READ(*,11) ASK
         UTEMP=TRUTH(ASK,'N')
      ENDIF

      WRITE(*,101)
      IF (.NOT.UTEMP) THEN

         WRITE (*,15) 'surface tension','dynes/cm',SIGMA
         READ(*,20) DUMMY
         IF (DUMMY.NE.0.) SIGMA=DUMMY

         WRITE (*,15) 'vapor pressure','dynes/cm**2',VP
         READ(*,20) DUMMY
         IF (DUMMY.GT.0.) VP=DUMMY

      ENDIF

      WRITE(*,67)
67    FORMAT(/' Do you want to include DEPOSITION [N] ? ',\)
      READ(*,11) ASK
      DODEPO=TRUTH(ASK,'N')

      IF (DODEPO) THEN

         WRITE (*,15) 'electric field','volts/cm',ELCFLD
         READ(*,20) DUMMY
         IF (DUMMY.GT.0.) ELCFLD=DUMMY
         IF (DUMMY.LT.0.) ELCFLD=0.

         WRITE (*,15) 'Ke dissipation','/sec',KE
         READ(*,20) DUMMY
         IF (DUMMY.GT.0.) KE=DUMMY

         WRITE (*,15) 'vessel diameter','cm',DVESSL
         READ(*,20) DUMMY
         IF (DUMMY.GT.0.) DVESSL=DUMMY

         WRITE (*,17)
```

```
  17     FORMAT(5X,'Charge approximation: 1 for singly charged,   ',
    $              '2 for Boltzmann')
         WRITE (*,18) ICHRG
  18     FORMAT (' ENTER desired charge approximation [',I1,'] : ',\)
         READ(*,21) IDUMMY
         IF (IDUMMY.GT.0) ICHRG=IDUMMY

      ENDIF

 15   FORMAT(' ENTER ',A,' in ',A,' [',1PE9.2,'] : ',\)
 16   FORMAT(' ENTER ',A,' [',1PE9.2,'] : ',\)
 20   FORMAT(G15.7)
 21   FORMAT(I1)


  90  WRITE(*,92)
  92  FORMAT(/' ENTER Enhancement FACTOR over Classical Nucleation ',
     1        '[1.] : ',\)
      READ(*,20) DUMMY
      IF (DUMMY.GT.0. .AND. DUMMY.NE.1.) THEN
         HIGHJ=.TRUE.
         FACTJ=DUMMY
      ENDIF

 100  WRITE(*,101)
 101  FORMAT(' ')
      WRITE (*,16) 'Relative Error Tolerance',RELERR
      READ(*,20) DUMMY
      IF (DUMMY.GT.0.) RELERR=DUMMY

      WRITE (*,16) 'Absolute Error Tolerance',ABSERR
      READ(*,20) DUMMY
      IF (DUMMY.GT.0.) ABSERR=DUMMY

      WRITE (*,16) 'Maximum T* Integrator Step Size',HMAXO
      READ(*,20) DUMMY
      IF (DUMMY.GT.0.) HMAXO=DUMMY

      IF (PRESET) RETURN

C----------------------------------------------------------------------

C          Find out what will change in these simulations

      WRITE(*,80)
  80  FORMAT('/Do you wish to do more than one simulation [N] ? ',\)
      READ(*,11) ASK
      REPEAT = TRUTH(ASK,'N')
      IF (REPEAT) THEN

         IF (USOR) THEN
            WRITE(*,81) 'Source Rates'
          ELSE
```

```
            WRITE(*,81) 'Initial Saturation Ratios'
         ENDIF
         READ(*,11) ASK
         NEWSR=TRUTH(ASK,'N')

         WRITE(*,81) 'Aerosol Number Concentrations'
         READ(*,11) ASK
         NEWAER=TRUTH(ASK,'N')

         WRITE(*,81) 'Aerosol Size'
         READ(*,11) ASK
         NEWSIZ=TRUTH(ASK,'N')

         WRITE(*,81) 'Temperatures'
         READ(*,11) ASK
         NEWTEM=TRUTH(ASK,'N')

      ENDIF
   81 FORMAT(' With Different ',A,' [N] ? ',\)

      IF (NODIM) THEN
         NDISP = 1
       ELSE
         WRITE(*,91) NDISP
   91    FORMAT(/'   Monitor Display Options:  9=None,  1=Nondim,  ',
     1          '2=Bimodal,  3=General'/
     2          ' Enter Choice [',I1,'] : ',\)
         READ(*,21) IDUMMY
         IF (IDUMMY.GT.0) NDISP=IDUMMY
         IF (NDISP.EQ.9) NDISP=0
      ENDIF

      RETURN
      END

C=====================================================================

      SUBROUTINE RASK2(NCASE,DIMSOR,FACTJ)

C  This Subroutine Interactively allows the user to change
C   one or more of a few parameters for parallel simulations
C   during a single execution of RSNM.

C  Changes:         DIMSOR or SRO, CINIT, DPI, TEMP

      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      LOGICAL REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /REPEAT/ REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      COMMON /INIT/ SRO,DIMNO,DIMDO,CINIT,DPI,TEMP,CS
```

```
 20  FORMAT(' ENTER ',A,' in ',A,' [',1PE9.2,'] : ',\)
 30  FORMAT(' ENTER ',A,' [',1PE9.2,'] : ',\)
 40  FORMAT(G15.7)
 50  FORMAT(I1)
101  FORMAT(' ')

     WRITE(*,101)
     IF ((NCASE.EQ.1 .OR. NEWSR) .AND. USOR) THEN
        WRITE(*,30) 'Dimensionless Source Rate',DIMSOR
        READ(*,40) DUMMY
        IF (DUMMY.GT.0.) DIMSOR = DUMMY
        IF (DUMMY.LT.-30.) GOTO 900
        IF (DUMMY.LT.0.) DIMSOR = 10.**DUMMY
     ENDIF

     IF (NCASE.EQ.1 .OR. (NEWSR .AND.(.NOT.USOR))) THEN
        WRITE(*,30) 'Initial Saturation Ratio', SR0
        READ(*,40) DUMMY
        IF (DUMMY.GT.0.) SR0=DUMMY
        IF (DUMMY.LT.0.) SR0=0.
        IF (DUMMY.LT.-30.) GOTO 900
     ENDIF

     IF (NCASE.EQ.1 .OR. NEWAER) THEN
        WRITE(*,20) 'Initial Aerosol Number','#/cc',CINIT
        READ(*,40) DUMMY
        IF (DUMMY.GT.0.) CINIT=DUMMY
        IF (DUMMY.LT.0.) CINIT=0.
     ENDIF

     IF ((NCASE.EQ.1 .OR. NEWSIZ).AND.CINIT.GT.0.) THEN
        DPIM = 1.E4 * DPI
        WRITE (*,20) 'Initial Aerosol Diameter','microns',DPIM
        READ(*,40) DUMMY
        IF (DUMMY.GT.0.) DPIM=DUMMY
        DPI = 1.E-4*DPIM
     ENDIF

     IF (UTEMP.AND. (NCASE.EQ.1 .OR. NEWTEM)) THEN
        WRITE(*,30) 'Temperature (C)',TEMP
        READ(*,40) DUMMY
        IF (DUMMY.GT.0.) TEMP=DUMMY
     ENDIF

     RETURN

 900 NCASE=-1
     RETURN
     END

C=================================================================

     SUBROUTINE RASK2D(NCASE,DIMSOR,DIMST,DIMKN)
```

```
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      LOGICAL REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /REPEAT/ REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /INIT/ SRO,DIMNO,DIMDO,CINIT,DPI,TEMP,CSAT
 30   FORMAT(' ENTER ',A,' [',1PE9.2,'] : ',\)
 40   FORMAT(G15.7)
101   FORMAT(' ')

      WRITE(*,101)
      IF ((NCASE.EQ.1 .OR. NEWSR) .AND. USOR) THEN
         WRITE(*,30) 'Dimensionless Source Rate',DIMSOR
         READ(*,40) DUMMY
         IF (DUMMY.GT.O.) DIMSOR = DUMMY
         IF (DUMMY.LT.-30.) GOTO 900
         IF (DUMMY.LT.O.) DIMSOR = 10.**DUMMY
      ENDIF

      IF (NEWTEM .OR. NCASE.EQ.1) THEN
         WRITE(*,30) 'Dimensionless Surface Tension',DIMST
         READ(*,40) DUMMY
         IF (DUMMY.GT.O.) DIMST=DUMMY

         WRITE(*,30) 'Dimensionless Transport Number',DIMKN
         READ(*,40) DUMMY
         IF (DUMMY.GT.O.) DIMKN=DUMMY
      ENDIF

      IF (NCASE.EQ.1 .OR. (NEWSR .AND.(.NOT.USOR))) THEN
         WRITE(*,30) 'Initial Saturation Ratio',SRO
         READ(*,40) DUMMY
         IF (DUMMY.GT.O.) SRO=DUMMY
         IF (DUMMY.LT.O.) SRO=0.
      ENDIF

      IF (NCASE.EQ.1 .OR. NEWAER) THEN
         WRITE(*,30) 'Dimensionless Initial Number',DIMNO
         READ(*,40) DUMMY
         IF (DUMMY.GT.O.) DIMNO=DUMMY
         IF (DUMMY.LT.O.) DIMNO=0.
      ENDIF

      IF (NCASE.EQ.1 .OR. NEWSIZ) THEN
         WRITE(*,30) 'Dimensionless Initial Diameter',DIMDO
         READ(*,40) DUMMY
         IF (DUMMY.GT.O.) DIMDO=DUMMY
      ENDIF

      RETURN

900   NCASE=-1
      RETURN
      END
```

```
C=====================================================================

      SUBROUTINE RSTEP (HO, HMAXO, HMAX, SCALES)

C     Figure out:  an appropriate printout stepsize TSTEP
C                      a maximum dimensionless time TMAX
C                    an appropriate initial stepsize HO

      REAL MW1
      INTEGER REGIME
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO,  SCALES
      COMMON /TIMES/ TSCALE,TSTEP,TMAX,TRES
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BETACE,GS,FIN1,FIN2,FACTJ
      COMMON /SNMX/ REGIME,DIAM1,CS,MW1,RSCAL1,TB,TS

      HMAX = HMAXO
      HO = 1.E-10


      IF (.NOT.NODIM.AND.TRES.GT.O.) THEN
         TMAX = TRES / TSCALE
         TSTEP = TMAX / 20.
       ELSEIF (TRES.LT.O.) THEN
         TMAX = -TRES
         TSTEP = TMAX / 20.
      ENDIF
      IF (PRESET) RETURN

      WRITE (*,16) 'Print-out T* Step Size',TSTEP
      READ(*,20) DUMMY
      IF (DUMMY.GT.O.) TSTEP=DUMMY
  16  FORMAT(' ENTER ',A,' [',1PE9.2,'] : ',\)
  20  FORMAT(G15.7)

      IF (.NOT.NOSHO) THEN
         TST = TSCALE * TSTEP
         IF (UHOURS) THEN
            TST = TST / 3600.
            WRITE(*,16) 'Equivalent TSTEP in Hours',TST
          ELSE
            WRITE(*,16) 'Equivalent TSTEP in Seconds',TST
         ENDIF
         READ(*,20) DUMMY
         IF (DUMMY.GT.O.) THEN
            TST = DUMMY
            IF (UHOURS) TST=TST*3600.
            TSTEP = TST / TSCALE
         ENDIF
      ENDIF

      IF (TSTEP.GT.O.) THEN
```

```
         TMAX = 500. * TSTEP
         GOTO 500
      ENDIF

C  TSTEP totally unknown . . . must guess appropriate value

      TSTEP = 0.5
      TMAX = 500.

      IF (SCALES) THEN

         IF (DIMST.LT.5.) THEN
            TSTEP=0.5*TSTEP
          ELSE IF (DIMST.GT.9) THEN
            TSTEP=4.*TSTEP
          ELSE IF (DIMST.GT.7.) THEN
            TSTEP=2.*TSTEP
         ENDIF

         IF (DIMSOR.GT.0.5) THEN
            TSTEP=8.*TSTEP
          ELSE IF (DIMSOR.GT.0.05) THEN
            TSTEP=4.*TSTEP
          ELSE IF (DIMSOR.GT.0.005) THEN
            TSTEP=2.*TSTEP
          ELSE
            TSTEP=0.5*TSTEP
         ENDIF

       ELSE

         IF (USOR) THEN
            TSTEP=TSTEP/DIMSOR
            HMAX=HMAXO/DIMSOR
          ELSE
            HMAX=HMAXO*TUSER
         ENDIF

         HO = 1.E-7 * TSTEP

      ENDIF


 500  UHOURS=.FALSE.
      IF (TSTEP*TSCALE .GT. 60.) UHOURS = .TRUE.

      RETURN
      END

C=====================================================================

      SUBROUTINE RSHOW (RELERR, NCASE)
```

```
C  Writes Out All Relevant Simulation Input Info to Unit 50

C  Writes Out Partial Description to Terminal

      PARAMETER ( AN = 6.02252E+23 )

      REAL MW, MW1, KE
      INTEGER REGIME
      LOGICAL CUT, CHECK, DEBUG, SAVSUM, SAVRUN, DOPLOT
      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO

      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      LOGICAL REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      CHARACTER*4 SPEED
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /REPEAT/ REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /OFLAGS/ NDISP,CUT,CHECK,DEBUG,SAVSUM,SAVRUN,DOPLOT
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BETACE,GS,FIN1,FIN2,FACTJ
      COMMON /SNMX/ REGIME,DIAM1,CS,MW1,RSCAL1,TB,TS
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     $                  DENSTY
      COMMON /DEPOST/ ELCFLD,KE,DI1,DVESSL,RHO,DIFF,ICHRG,TGAS
      COMMON /EPCOMY/ YMIN,HMAX
      COMMON /RESULT/ DIMT,DIMN,DIMM,DIMA,DIMD,TIME,TOTN,TOTM,TOTA,DPX
      COMMON /INIT/ SR0,DIMNO,DIMDO,CINIT,DPI,TEMP,CSAT

      WRITE(50,5)
    5 FORMAT(' ----------------------------------------',
     1       '-------------------------------------')

      IF (NCASE.EQ.1) THEN

         WRITE(*,10)  DOKELV,USEBCE,SCALES,HIGHJ,LOTHE,DODEPO,REGIME
         WRITE(50,10) DOKELV,USEBCE,SCALES,HIGHJ,LOTHE,DODEPO,REGIME
   10    FORMAT(/' DOKELV=',L1,'  USEBCE=',L1,'  SCALES=',L1,
     $  '  HIGHJ=',L1,'  LOTHE=',L1,'  DODEPO=',L1,'  REGIME=',I1/)

         WRITE(*,20)  FIN1,FIN2,GS,RELERR,YMIN,HMAX
         WRITE(50,20) FIN1,FIN2,GS,RELERR,YMIN,HMAX
   20    FORMAT(' Fin=',2F6.3,2X,'GS=',F5.0,2X,'ERROR=',1PE9.2,2X,
     %     'YMIN=',1PE9.2,2X,'HMAX=',1PE9.2 /)

         IF (HIGHJ.AND.FACTJ.NE.1.) THEN
            IF (FACTJ.GT.1.) SPEED='FAST'
            IF (FACTJ.LT.1.) SPEED='SLOW'
            WRITE(*,30)  FACTJ,SPEED
            WRITE(50,30) FACTJ,SPEED
   30       FORMAT(' Nucleation is ',1PE12.3,2X,A4,/)
         ENDIF

         IF (.NOT.NODIM) THEN
            WRITE(*, 56) SIGMA,VP,DENSTY,MW,BCE
```

```
          WRITE(50,56) SIGMA,VP,DENSTY,MW,BCE
 56       FORMAT(' Surface Tension = ',F7.3,' dynes/cm',6X,
   1               ' Vapor Pressure = ',1PE9.3,' dynes/cm**2' /
   2               ' Density = ',0PF6.3,' g/cc',6X,
   3               ' MW = ',0PF7.2,6X,'Bce = ',F7.3 /)
          VPAT = VP / 1.0133E6
          RMASS = CS * MW / AN
          WRITE(*, 58) VPAT,CS,1.E12*RMASS
          WRITE(50,58) VPAT,CS,1.E12*RMASS
 58       FORMAT(' Psat = ',1PE10.2,' Atm',5X,'Nsat = ',1PE10.2,
   1               ' /cc',5X,'Msat = ',1PE10.2,' ug/m**3' /)
       ENDIF

       IF (DODEPO) THEN
         WRITE (50,60) ELCFLD,KE
 60      FORMAT ('  Electric Field = ',F9.3,' V/cm',6x,
   1              'Ke = ',1PE9.3,' per second')
         IF (ICHRG.EQ.1) THEN
            WRITE (50,71)
 71         FORMAT(' Assuming singly charged aerosol.'/)
          ELSE
             WRITE (50,72)
 72          FORMAT(' Assuming Boltzmann charge on aerosol.'/)
         ENDIF
       ENDIF

       IF (.NOT.NODIM) THEN
       IF (SCALES) THEN
          WRITE(*, 90) TS,TS/3600.
          WRITE(50,90) TS,TS/3600.
 90       FORMAT(' Time T* Scaled to Source Time =',
   1               1PE10.3,' Seconds  = ',0PF9.3,' Hours.'/)
        ELSE
          WRITE(*, 91) TB,TB/3600.
          WRITE(50,91) TB,TB/3600.
 91       FORMAT(' Time T* Scaled to Collisional Time =',
   1                  1PE10.3,' Seconds  = ',0PF9.3,' Hours.'/)
       ENDIF
       ENDIF

    ENDIF

    WRITE (*,80) DIMSOR,DIMST,DIMKN,SR0
    WRITE(50,80) DIMSOR,DIMST,DIMKN,SR0
 80 FORMAT (2X,'Rs* = ',1PE9.3,3X,'ST* = ',0PF7.3,3X,
   1          'Kn* = ',1PE9.3,3X,'S0 = ',0PF8.2 /)
    IF (.NOT.NODIM) THEN
       WRITE(*, 81) CINIT,1.E4*DPI
       WRITE(50,81) CINIT,1.E4*DPI
 81    FORMAT(' Initial Number= ',1PE10.3,' per cc',6x,
   1   'Initial Dp= ',0PF9.4,' um'/)
    ENDIF
```

```
c     WRITE(*,100) GS
c     WRITE(*,110) DIMSOR,DIMST
c     IF (USEBCE) WRITE(*,120) BETACE,DIMKN
c     IF (.NOT.USEBCE) WRITE(*,130) DIMKN
c100  FORMAT(/15X,'***  CONDENSING SYSTEM PROPERTIES  ***',5X,
c    1  'gs=',F5.0 /)
c110  FORMAT(' Dimensionless Source Rate =',1PE10.3,4X,
c    1  'Dimensionless Surface Tension =',0PF7.3 /)
c120  FORMAT(' Dimensionless Diffusivity =',F7.3,5X,
c    1  'Dimensionless Knudsen Number =',F8.2 /)
c130  FORMAT(' Dimensionless Knudsen Number =',1PE11.2,5X,
c    1  'For Modified Fuchs-Sutugin' /)

      RETURN
      END

C=================================================================

      SUBROUTINE ROUT (TDIM)

C  Displays intermediate calculations on terminal

      PARAMETER ( F13 = 0.3333333, F23 = 0.6666667 )
      PARAMETER ( AN = 6.02252E+23 )
      REAL MW, MW1, KE
      INTEGER REGIME
      LOGICAL CUT, CHECK, DEBUG, SAVSUM, SAVRUN, DOPLOT
      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      LOGICAL REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /REPEAT/ REPEAT, NEWSR, NEWAER, NEWSIZ, NEWTEM
      COMMON /OFLAGS/ NDISP,CUT,CHECK,DEBUG,SAVSUM,SAVRUN,DOPLOT
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BETACE,GS,FIN1,FIN2,FACTJ
      COMMON /SNMX/ REGIME,DIAM1,CS,MW1,RSCAL1,TB,TS
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     $               DENSTY
      COMMON /DEPOST/ ELCFLD,KE,DI1,DVESSL,RHO,DIFF,ICHRG,TGAS
      COMMON /EPCOMY/ YMIN,HMAX
      COMMON /RESULT/ DIMT,DIMN,DIMM,DIMA,DIMD,TIME,TOTN,TOTM,TOTA,DPX
      COMMON /TIMES/ TSCALE,TSTEP,TMAX,TRES
      COMMON /XARRAY/ X(5)
      EQUIVALENCE (X(1),S)
      EQUIVALENCE (X(2),RN1) , (X(3),RM1)
      EQUIVALENCE (X(4),RN2) , (X(5),RM2)

      DIMN = RN1 + RN2
      DIMM = RM1 + RM2
      DIMA = FIN1*RN1**F13*RM1**F23 + FIN2*RN2**F13*RM2**F23
      IF (DIMN.GT.0.) THEN
         GBAR=DIMM/DIMN
         DIMD=GBAR**F13
         DPBAR=DIAM*DIMD
       ELSE
         GBAR=0.
         DIMD=0.
      ENDIF

      RDP1=0.
      RDP2=0.
      IF (RN1.GT.0.) RDP1=(RM1/RN1)**F13
      IF (RN2.GT.0.) RDP2=(RM2/RN2)**F13

      DIMC = -1.
      IF (S.GT.1.) DIMC = DIMST / ALOG(S)

      IF (NODIM) GOTO 100

      TIME = TDIM * TSCALE
      IF (UHOURS) TIME = TIME / 3600.

      TOTN = CS * DIMN

      DCR = -1.E-4
      IF (S.GT.1.) DCR = DIKELV / ALOG (S)
      DPBAR=DIAM*DIMD
      DP1=RDP1*DIAM
      DP2=RDP2*DIAM
      TOTN1=RN1*CS
      TOTN2=RN2*CS

C  Since we may want a single Dp, use value of larger number mode

      IF (TOTN1.GE.TOTN2) THEN
         DPX = DP1
       ELSE
         DPX = DP2
      ENDIF

C  Dimensional Totals, #/cc and g/cc

      TOTN = CS * DIMN
      TOTM = CS * DIMM * MW / AN

 100  IF (SAVRUN) THEN
         IF (NOSHO) THEN
            WRITE(70,610) DIMT,S,DIMN,DIMM,DIMD,DIMC
          ELSE
            WRITE(70,630) DIMT,S,DIMN,DIMM,TIME,TOTN,DPX,DCR
         ENDIF
      ENDIF


      IF (NDISP.EQ.1) THEN
         WRITE(*,610) DIMT,S,DIMN,DIMM,DIMD,DIMC
 610      FORMAT(1X,F10.3,1X,F10.4,1X,1PE12.4,1X,
     1          1PE12.4,1X,0PF9.2,1X,0PF9.2)

       ELSEIF (NDISP.EQ.2) THEN
          WRITE (*,620) TIME,S,DP1,TOTN1,DP2,TOTN2,TOTM
 620      FORMAT(1X,1PE10.3,1X,0PF8.3,1X,4PF9.4,1X,1PE10.3,
     1           2X,4PF9.4,1X,1PE10.3,12PF12.3)

       ELSEIF (NDISP.EQ.3) THEN
          WRITE(*,630) DIMT,S,DIMN,DIMM,TIME,TOTN,DPX,DCR
 630      FORMAT(1X,F9.3,1X,F9.4,1X,1PE9.2,1X,1PE9.2,1X,
     1          0PF9.3,1X,1PE10.3,1X,4PF8.4,1X,4PF8.4)

      ENDIF

c------------------------ old formats -----------------------------
c610      FORMAT(' T*=',F9.3,'  S=',F9.4,'  N*=',1PE11.4,
c    1          '  M*=',1PE11.4,'  D*=',0PF8.2,'  C*=',0PF8.2)
c620      FORMAT(' T*=',F9.3,'  S=',F9.4,'  N*=',1PE11.4,
c    1          '  t=',0PF10.3,1X,A1,2X,'  Np=',1PE10.3,
c    2          '  Dp=',4PF9.4,'  Dc=',4PF9.4)
c         WRITE (*,630) TIME,CHT,S,DP1,DP2,TOTN1,TOTN2
c630      FORMAT (' t=',1PE10.3,1X,A1,2X,'S=',0PF7.3,2X,'Dp=',4P2F10.4,
c    1       2x,'Ni=',1P2E10.3)
c
c      IF (.NOT.SCALES) WRITE(*,181) TIME,CHT,S,DIMN,DIMM,DIMD
c181  FORMAT(' t=',1PE11.3,1X,A1,2X,'SR=',0PF7.3,2X,'N*=',1PE11.3,2X,
c    $          'M*=',E11.3,2X,'D*=',0PF12.1)
C                                DIMT,S,DIMN,DIMM,DIMD,DPBAR,DCR,TOTN
C 77  FORMAT(' T*=',F9.3,'  S=',F9.4,'  N*=',1PE11.4,
C    $      '  M*=',1PE11.4,:,'  D*=',1PE11.4,
C    $      '  Dp=',4PF9.4,'  Dc=',4PF9.4,'  Np=',1PE11.4)
c------------------------------------------------------------------

      IF (DOPLOT) THEN
         PHI2=0.
         RC=(RC1+RC2)
         IF (RMJ+RMC.GT.0.) PHI2=RJ/(RJ+RC/GS)
         PHI3=RC+RJ*GS
         DIMJ=RJ*GS
C  Mass Nondimensionalization= PHI2*PHI3
      ENDIF
C     WRITE(23,85) DIMT,S,DIMA,DIMJ,PHI2,PHI3
C  85 FORMAT(' T=',F8.3,2X,'SR=',F7.3,2X,'A=',1PE10.2,2X,'J=',
C    $ E10.2,2X,'PHI=',2E11.2)
      IF (DOPLOT) THEN
         WRITE(99,99) TIME,DIMT,S,DIMA,DIMN,DIMM,DIMJ,-1.,PHI2,PHI3,RNJ
   99    FORMAT(1PE11.3,0P2F11.4,1P4E11.3,0PF11.6,1P3E11.3)
      ENDIF

      RETURN
      END
```

```
C=====================================================================

      SUBROUTINE HEADER (NDISP,UHOURS)

      LOGICAL UHOURS
      CHARACTER*7 LABEL

      IF (UHOURS) THEN
         LABEL = ' Hours '
       ELSE
         LABEL = 'Seconds'
      ENDIF

      IF (NDISP.EQ.1) THEN
         WRITE(*,100)
 100     FORMAT(6X,'Time',8X,'S',8X,'Number',7X,'Mass',6X,
     1           'Dp(ModeN)',4X,'Dcrit'/
     2           7X,'*',9X,' ',11X,'*',11X,'*',11X,'*',13X,'*')
       ELSEIF (NDISP.EQ.2) THEN
         WRITE(*,200) LABEL
 200     FORMAT(5X,'Time',7X,'S',5X,'Dp(Mode1)',2X,'Np(Mode1)',2X,
     1           'Dp(Mode2)',2X,'Np(Mode2)',4X,'Mass'/
     2           3X,A7,6X,' ',6X,'Microns',4X,'Microns',6X,
     3           '#/cc',6X,'#/cc',6X,'ug/m**3')
       ELSEIF (NDISP.EQ.3) THEN
         WRITE(*,300) LABEL
 300     FORMAT(5X,'Time',7X,'S',6X,'Number',5X,'Mass',6X,
     1           'Time',5X,'Number',3X,'Dp(ModeN)',3X,'Dcrit'/
     2           6X,'*',9X,' ',8X,'*',9X,'*',7X,A7,4X,'#/cc',5X,
     3              'Microns',2X,'Microns')
      ENDIF

      RETURN
      END

C=====================================================================

      SUBROUTINE RSUM (S)

      PARAMETER ( AN = 6.02252E+23 )
      LOGICAL CUT, CHECK, DEBUG, SAVSUM, SAVRUN, DOPLOT
      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      REAL MW
      CHARACTER*3 TLAB
      EQUIVALENCE (X(2), RN1) , (X(4), RN2)
      COMMON /XARRAY/ X(5)
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /OFLAGS/ NDISP,CUT,CHECK,DEBUG,SAVSUM,SAVRUN,DOPLOT
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BETACE,GS,FIN1,FIN2,FACTJ
      COMMON /RESULT/ DIMT,DIMN,DIMM,DIMA,DIMD,TIME,TOTN,TOTM,TOTA,DPX
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     1                     DENSTY
      COMMON /INIT/ SRO,DIMNO,DIMDO,CINIT,DPI,TEMP,CS
      COMMON /CASE/ NCASE,RNJO

      IF (DIMNO.EQ.O.) RNJO = DIMN
      TLAB = 'sec'
      IF (UHOURS) TLAB = 'hr '

      IF (NDISP.EQ.O) THEN
         WRITE(*,220) TIME,S,DIMN,DIMM,DIMD
 220     FORMAT(' t =',1PE11.3,2X,'SR=',0PF7.3,2X,'N*=',1PE11.3,2X,
     1          'M*=',E11.3,2X,'D*=',0PF12.1)
      ENDIF

C  None of the summary files now save final S, Time, or T*

C          General Compact Nondimensional Summary (SUM.)

      IF (USOR) THEN
         WRITE(60,610) DIMSOR,DIMST,DIMKN,DIMNO,DIMDO,DIMN,DIMD
 610     FORMAT(1X,1PE10.2,0PF7.3,1P5E11.3)
       ELSE
         WRITE(60,620) SRO,DIMST,DIMKN,DIMNO,DIMDO,DIMN,DIMD
 620     FORMAT(1X,0PF10.2,0PF7.3,1P5E11.3)
      ENDIF

C          Verbose Summary (RSNM.)

      IF (USOR) THEN
         IF (NODIM) THEN
            WRITE(*, 510) DIMSOR,DIMNO,DIMN,DIMT
            WRITE(50,510) DIMSOR,DIMNO,DIMN,DIMT
 510        FORMAT(' R*=',1PE9.2,' & Ni*=',1PE10.3,' result in ',
     1             'N*=',1PE10.3,' by t*=',1PE9.2)
          ELSE
            WRITE(*, 520) DIMSOR,CINIT,TEMP,TOTN,TIME,TLAB
            WRITE(50,520) DIMSOR,CINIT,TEMP,TOTN,TIME,TLAB
 520        FORMAT(' R*=',1PE9.2,' & ',1PE10.3,' /cc at',0PF6.1,
     1          ' C give N=',1PE10.3,' /cc by t=',1PE8.1,' ',A3)
         ENDIF
       ELSE
         IF (NODIM) THEN
            WRITE(*, 530) SRO,DIMNO,DIMN,DIMT
            WRITE(50,530) SRO,DIMNO,DIMN,DIMT
 530        FORMAT(' SO=',0PF7.2,' & Ni*=',1PE10.3,' give ',
     1             'N*=',1PE10.3,' by t*=',1PE9.2)
          ELSE
            WRITE(*, 540) SRO,CINIT,TEMP,TOTN,TIME,TLAB
            WRITE(50,540) SRO,CINIT,TEMP,TOTN,TIME,TLAB
 540        FORMAT(' SO=',0PF7.2,' & ',1PE10.3,' /cc at',0PF6.1,
     1        ' C give N=',1PE10.3,' /cc by t=',1PE9.2,' ',A3)
         ENDIF
      ENDIF
      WRITE(*,*) ' '

      IF (.NOT.SAVSUM) RETURN

C           General Summary with Dimensional Values (SUM2.)

C  RSMASS is condensable mass source rate in ug/m**3/hr
C  VPAT is consdensible saturation vapor pressure in atm

      IF (.NOT.NODIM) THEN
        RSMASS = 1.E12*3600.*DIMSOR*RSCALE*MW/AN
        VPAT = VP / 1.013E6
        IF (USOR) THEN
          WRITE(80,810) RSMASS,SIGMA,VPAT,TEMP,CINIT,DPI,TOTN,DPX
 810      FORMAT(1X,1PE10.3,0PF7.2,1PE11.3,0PF7.2,1PE11.3,4PF7.4,
     1               1PE11.3,4PF8.4)
         ELSE
          WRITE(80,820) SRO,SIGMA,VPAT,TEMP,CINIT,DPI,TOTN,DPX
 820      FORMAT(1X,0PF10.2,0PF7.2,1PE11.3,0PF7.2,1PE11.3,4PF7.4,
     1               1PE11.3,4PF8.4)
        ENDIF
      ENDIF

C           Summary of Suppression by Initial Aerosol (SUPPRESS.)

      IF (USOR) THEN
         IF (NOSHO) THEN
            WRITE(90,910) DIMSOR,DIMDO,DIMNO,DIMN,
     1            RN1/RNJO,RN2/RNJO,DIMN/(RN1+RNJO),DIMN/RNJO
 910        FORMAT(1X,1PE10.3,0PF7.2,6(1PE10.3))
          ELSE
            WRITE(90,920) DIMSOR,1.E4*DPI,CINIT,TOTN,
     1         RN1/RNJO,RN2/RNJO,DIMN/(RN1+RNJO),DIMN/RNJO
 920        FORMAT(1X,1PE10.3,0PF4.0,F6.3,6(1PE10.3))
         ENDIF
       ELSE
         IF (NOSHO) THEN
            WRITE(90,930) SRO,DIMDO,DIMNO,DIMN,
     1            RN1/RNJO,RN2/RNJO,DIMN/(RN1+RNJO),DIMN/RNJO
 930        FORMAT(1X,F4.0,F7.2,6(1PE11.3))
          ELSE
            WRITE(90,940) SRO,1.E4*DPI,CINIT,TOTN,
     1         RN1/RNJO,RN2/RNJO,DIMN/(RN1+RNJO),DIMN/RNJO
 940        FORMAT(1X,F4.0,F6.3,6(1PE11.3))
         ENDIF
      ENDIF

      RETURN
      END

C=====================================================================
      FUNCTION TRUTH(ASK,DASK)

C  'Y' or 'y' = .TRUE. ; 'N' or 'n' = .FALSE. ; ASK defaults to DASK

      LOGICAL TRUTH
      CHARACTER*1 ASK,DASK

      IF (DASK.EQ.'Y'.OR.DASK.EQ.'y') THEN
         TRUTH=.TRUE.
      ELSE
         TRUTH=.FALSE.
      ENDIF

      IF (ASK.EQ.'Y'.OR.ASK.EQ.'y') TRUTH=.TRUE.
      IF (ASK.EQ.'N'.OR.ASK.EQ.'n') TRUTH=.FALSE.

      RETURN
      END
```

```
$DEBUG
C-----------------------------------------------------------------------
C   Derivative Package for Dual Mode S-N-M Model
C       Classical B-D-Z or Lothe-Pound Homogeneous Nucleation
C       Modified Fuchs-Sutugin or Chapmann-Enskog Condensation
C        with Finagle Factors to account for polydispersity
C       First Mode is Primary (pre-existing) Aerosol,
C       Second Mode is Homogeneously Nucleated Aerosol.
C-----------------------------------------------------------------------
C
      SUBROUTINE DIFFUN(NEQ,TIME,X,DXDT)

      PARAMETER ( PI = 3.141593 )
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. , THREE=3. )
      PARAMETER ( F23 = 0.66666667 , F13 = 0.33333333 )
C                           Boltzmann Constant, erg/K/molecule
      PARAMETER ( BK = 1.38054E-16 )
C                           Avogadro's Number, molecules/mole
      PARAMETER ( AN = 6.02252E+23 )
C                           Planck's Constant, erg-sec
      PARAMETER ( HP = 6.6256E-27 )

      EQUIVALENCE (RDK, DIMST)
C                    Reduced Kelvin Diameter, DIKELV/D1 (=SUE/1.5)
C       Note Length Scaling is to monomer diameter:
C       RDP=dp/d1   DIMKN=2.*MFPa/d1   RDK=dkelvin/d1=DIMST
C       For consistency, remember DIMKN=6*DIFFUS/(DIAM*VEL1)
C       Code calculates true Kn = D1/Dp  *  DIMKN / (3.*BCE)
C                    adjusted Kn = D1/Dp  *  DIMKN
C

      REAL X(5),DXDT(5)

      REAL MW
      INTEGER REGIME

      LOGICAL CHECK, DEBUG
      LOGICAL DOKELV,USEBCE,SCALES,HIGHJ,LOTHE,DODEPO
C
C  /DFLAGS/ contains the major flags for alternate physical
C  process assumptions.
C
      COMMON /DFLAGS/ DOKELV,USEBCE,SCALES,HIGHJ,LOTHE,DODEPO
C
C    *  DOKELV -- includes the Kelvin effect on condensation
C    *  DODEPO -- includes electrostatic deposition
C    *  USEBCE -- uses Chapmann Enskog instead of modified Fuchs-Sutugin
C    *  SCALES -- use TS (source) rather than TB (collision) time scaling
C    *  HIGHJ  -- artificially boost nucleation rate by FACTJ
C    *  LOTHE  -- use Lothe-Pound nucleation rate expression
C
C
```

```
C  /SNM2/ contains basic dimensionless properties of
C   the aerosol system which are needed to calculate
C   the scaled rates of condensation and nucleation.
C        BCE is irrelevant for Modified Fuchs-Sutugin.
C
C
      COMMON /SNM2/ DIMSOR,DIMST,DIMKN,BCE,GS,FIN1,FIN2,FACTJ
C
C
C  /SNMX/ contains physical process options flags
C   and a few dimensional properties needed for L-P nucleation.
C   It could be removed if only normal Classical Nucleation
C   is being done, with no deposition (needs time scaling).
C
      COMMON /SNMX/ REGIME,DIAM,CS,MW,RSCALE,TB,TS
C
C
C  /RPASS/ contains properties being calculated in DIFFUN and
C   being passed for the benefit of print-out lines in RSNM.
C
      COMMON /RPASS/ SS1,SS2,AKN1,AKN2,RDP1,RDP2,RNRJ,RNRC1,RNRC2
C
C
C
C                    First Call prints out input values
      DATA CHECK /.FALSE./
C
C                    All Calls print out calculated values
      DATA DEBUG /.FALSE./
C
C
C***                 Use our mneumonics for dependent variables
C
C                                Saturation Ratio
      S=X(1)
C                                Reduced Primary Number Concentration
      RN1=X(2)
C                                Reduced Primary Mass Concentration
      RM1=X(3)
C                                Reduced Secondary Number Concentration
      RN2=X(4)
C                                Reduced Secondary Mass Concentration
      RM2=X(5)
C
C
C  Will these defaults cause numerical problems for integrator?
C
C                                Default Reduced Primary Diameter
      RDP1=GS**F13
C                                Default Reduced Secondary Diameter
      RDP2=RDP1
C
      IF (RN1.GT.ZERO.AND.RM1.GT.ZERO) RDP1=(RM1/RN1)**F13
      IF (RN2.GT.ZERO.AND.RM2.GT.ZERO) RDP2=(RM2/RN2)**F13
C
C
      IF (CHECK) THEN
        WRITE(*,200) S,RN1,RN2,RM1,RM2
        WRITE(*,210) DIMSOR,DIMST,DIMKN,BCE,GS
C                            Only Print This Out Once
        CHECK=.FALSE.
  200   FORMAT(/' S=',F7.3,3X,'rN=',1P2E11.2,3X,'rM=',1P2E11.2 /)
  210   FORMAT(' DIMSOR=',1PE10.2,3X,'DIMST=',0PF7.3,3X,'mon Kn=',
     $      F7.1,3X,'BCE=',F7.3,3X,'GS=',F8.1 /)
      END IF
C
C
C-------------------------------------------------------------------------
C
C***    Calculate Dimensionless Rates of Nucleation and Condensation
C
C***              Nondimensionalized to Saturated Collision Rate
C
C***                        Classical Nucleation Theory
C
      IF (S.GT.ONE) THEN
         WCR= 0.5 * DIMST ** 3 / (ALOG(S) **2)
         RJ=S*S*SQRT(DIMST/6./PI)*EXP(-WCR)
C               Also, For Classical BDZ:
C        GC= ( DIMST / ALOG(S)) ** 3
C        ZELD= SQRT(WCR/3./PI) / GC
C        RJ=S*S*(GC**F23)*ZELD*EXP(-WCR)
         IF (LOTHE) THEN
            XLP=1.0
            XFACT=4.*(ALOG(S)**2)/(DIMST**3)
            DO 100 I=1,10
 100           XLP = 1.0 - XFACT / (XLP**2)
            GCL=(DIMST/ALOG(S))**3
            GLP=GCL*XLP**3
            WLP=0.5*DIMST*GLP**(2./3.)
            WCL=0.5*DIMST*GCL**(2./3.)
            ZCL=SQRT(WCL/3./PI)/GCL
            ZLP=ZCL*SQRT(1.-3.*(1.-XLP))
            SLOP=ALOG(1.1E-5)+12.*ALOG(XLP*DIMST/ALOG(S))
            SLOP=SLOP-ALOG(CS)+3.*ALOG(6.*MW/PI)
            SLOP=SLOP+3.*(ALOG(DIAM)+ALOG(BK*T)-ALOG(AN)-2.*ALOG(HP))
C                                     = CLS/CS
C                                     * RSCALE for #/cc/sec
CD        RJC=ZCL*S*S*(GCL**(2./3.))*EXP(-WCL)
C                                     * RSCALE for #/cc/sec
           RJ=ZLP*S*(GLP**(2./3.))*EXP(SLOP-WLP)
CD        WRITE(*,333) RJC,RJ,XLP,GLP,ZLP,WLP,SLOP
CD333    FORMAT(' RJC=',1PE11.3,'  RJ=',1PE11.3,'  GLP=',1PE11.3 /
CD       1  '  XLP=',0PF8.4,'  ZLP=',F8.4,'  WLP=',1PE11.4,'  SL=',E11.4)
         END IF
      ELSE
```

```
          RJ=ZERO
      END IF
C
C  Optionally scale nucleation rate up or down
C
      IF (HIGHJ) RJ=FACTJ*RJ
C
C-------------------------------------------------------------------
C
C  CALCULATE CONDENSATIONAL DRIVING FORCE (SUPERSATURATION)
C
      SS=S-1.
C
C  NORMALLY CORRECT FOR THE KELVIN EFFECT
C
C          (AMIN1 is used to avoid Overflow)
C
      IF (DOKELV) THEN
         DRAT=AMIN1(RDK/RDP1,10.)
         SS1=S-EXP(DRAT)
         DRAT=AMIN1(RDK/RDP2,10.)
         SS2=S-EXP(DRAT)
       ELSE
         SS1=SS
         SS2=SS
      END IF
      IF (SS1.LT.0.) SS1=0.
      IF (SS2.LT.0.) SS2=0.
C
C
C                        McMurry's Condensation Rate
C
C        Calculate Condensation Rate onto Monodisperse Aerosol
C        Note that for given N and M, a monodisperse distribution
C         maximizes condensation.  True rate would be lower generally.
C
C                             Effective Primary Knudsen Number
      EKN1= DIMKN / RDP1
C                             Effective Secondary Knudsen Number
      EKN2= DIMKN / RDP2

      IF (USEBCE) THEN
C                             True Knudsen Number = DIMKN/(3.*BCE*RDP)
         AKN1= EKN1 / (3.*BCE)
         AKN2= EKN2 / (3.*BCE)
         FK1=FKCE(AKN1,BCE)
         FK2=FKCE(AKN2,BCE)
       ELSE
         FK1=FKFS(EKN1)
         FK2=FKFS(EKN2)
      END IF

      IF (REGIME.EQ.1) THEN
```

```
C                                          FREE MOLECULE REGIME assumed
         FK1=1.
         FK2=1.

       ELSE IF (REGIME.EQ.2) THEN
C                                          CONTINUUM REGIME assumed
         FK1=4./3.*EKN1
         FK2=4./3.*EKN2

      END IF

      RC1=FIN1*SS1*RN1*(RDP1**2)*FK1
      RC2=FIN2*SS2*RN2*(RDP2*RDP2)*FK2


C-----------------------------------------------------------------------

C       Generalized Collision Function (Coagulation or Condensation)
C       where FKij is based on Knudsen Number of larger particle (?)
C  RNAij=FIN*RNi*RNj*FKij/DIMSOR*(SQRT(RDi**-3+RDj**-3)*(RDi+RDj)**2)
C
C  Value of Braced Group for Various Simple Cases
C       i=j=1            => sqrt(32)     and collisions are usually ineffective
C       i=1 , j>>1       => RDj**2       standard condensational growth
C       i>>1, j>>i       => RDj**2/RDi**1.5
C       i=j , j>>1       => sqrt(32*RDj) self-coagulation!
C
C      RA11=FIN1*FK1/DIMSOR*RN1*RN1*SQRT(32.*RDP1)
C      RA22=FIN2*FK2/DIMSOR*RN2*RN2*SQRT(32.*RDP2)
C      RA12=(FIN1*FIN2)*AMIN1(FK1,FK2)/DIMSOR*RN1*RN2*
C     % SQRT(1./RDP1**3+1./RDP2**3)*(RDP1+RDP2)**2
C       Note:  subtract RA12 from mode with larger RDP
C
C       *** CURRENTLY COAGULATION IS NOT INCLUDED
C       *** ITS EFFECT WOULD BE NEGLIGIBLE
C
C-----------------------------------------------------------------------

C  Now calculate deposition coefficients, using Pete McMurry's theory,
C     which includes electrostatics.

      IF (DODEPO) THEN
         IF (SCALES) THEN
            TSCALE=TS
          ELSE
            TSCALE=TB
         ENDIF

         BDEP1 = DEPSIT(RDP1,AKN1)
         BDEP2 = DEPSIT(RDP2,AKN2)
         BET1 = BDEP1*TSCALE
         BET2 = BDEP2*TSCALE
C
```

```
C  Debugging lines for DEPOST calculations follow:
C
c            write (63,63) time,3600.*bdep1,3600.*bdep2,bet1,bet2
c 63         format(' time=',1pe10.2,'  bdep1=',1pe10.2,
c    $         '  bdep2=',1pe10.2,'  bet1=',e10.2,'  bet2=',e10.2)
C
        ELSE
          BET1=0.
          BET2=0.
       END IF

C
C-----------------------------------------------------------------------
C
C***    Rescale Dimensionless Rates to RS (TS) from RB (TB)
C          Note DIMSOR = TB / TS
C
       IF (SCALES) THEN
          RJ  = RJ / DIMSOR
          RC1 = RC1 / DIMSOR
          RC2 = RC2 / DIMSOR
          DXDT(1) = 1.
        ELSE
          DXDT(1) = DIMSOR
       END IF
c      WRITE(99,*) 'Rs = ',RJ,RC1,RC2
C
c      IF (RJ.LT.1.E-30) RJ = 0.
C
C***             Calculate Derivatives based on Dimensionless Number Fluxes
C

C                                                 reduced dS/dt
      DXDT(1) = DXDT(1) - GS*RJ - RC1 - RC2

C                                                 reduced dN1/dt
      DXDT(2) = 0. - BET1*RN1

C                                                 reduced dM1/dt
      DXDT(3) = RC1 - BET1*RM1

C                                                 reduced dN2/dt
      DXDT(4) = RJ - BET2*RN2

C                                                 reduced dM2/dt
      DXDT(5) = GS*RJ + RC2 - BET2*RM2


      IF (DEBUG.AND.(RN1+RN2).GT.ZERO) THEN
C                                                 nucleation fraction
         PHI=RNRJ/(RNRJ+RNRC2/GS)
         WRITE(*,250) S,SS2,AKN2,RDP2,PHI
250      FORMAT(' S=',F7.3,3X,'SS=',F7.3,3X,'KN2=',F9.4,3X,
     $      'RDP2=',F10.2,3X,'PHI=',1PE10.2)

      END IF

      RETURN
      END

C==================================================================

        FUNCTION FKCE(KN,BCE)

C          Flux Ratio, Chapmann & Enskog / Kinetic Limit

C       For Kn>>1., FK = 1.               For Kn<<1., FK = 4.*BCE*KN << 1.
C        Kinetic Limit                     Continuum Limit
C
C          Knudsen Number based on true monomer mean free path
C
        REAL KN
C
        T1 = 4. * ( BCE * KN ) ** 2
        T2 = 2.88 * BCE * BCE * KN
        FKCE = ( T1 + T2 ) / ( T1 + T2 + 0.52*BCE*KN + 0.72*BCE )
        RETURN
        END

C==================================================================

        FUNCTION FKFS(EKN)

C          Flux Ratio, Fuchs & Sutugin / Kinetic Limit

C          Knudsen Number defined as 6 Diffus / (Vel1 * Dp)
C                  = 3. * BCE * KN(true)

        REAL EKN
        PARAMETER (F43 = 1.33333333 )
        FKFS = F43*EKN*(1.+EKN)/(1.+1.71*EKN+F43*EKN*EKN)
        RETURN
        END
      SUBROUTINE DIMSET(DIMST,DIMKN,TB,CS)
C
C***********************************************************************
C
C  PURPOSE:     To Set Dimensional and Dimensionless Physical Parameters
C
C  ON ENTRY:
C
C  ON RETURN:
C       DIMST   Dimensionless Surface Tension
C       DIMKN   Dimensionless Monomer Knudsen Number
C       TB      Characteristic Collision Time (Saturated Vapor) [sec]
C       CS      Saturated Vapor Concentration [#/cc]
C       /DS/    all COMMON variables set (dimensional properties)
C
C  COMMENTS:
C       This subroutine must be specialized for each compound.
C
C***********************************************************************
C
      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. , THREE=3. )
      PARAMETER ( TH1=0.3333333 , TH2=0.6666667 )
C                           Gas Constant, erg/K/mole
      PARAMETER ( RGAS = 8.31433E+7 )
C                           Boltzmann Constant, erg/K/molecule
      PARAMETER ( BK = 1.38054E-16 )
C                           Avogadro's Number, molecules/mole
      PARAMETER ( AN = 6.02252E+23 )
      REAL MW
C
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     $                  DENSTY
      COMMON /DIMDEP/ T
C
C-----------------------------------------------------------------------
C
C                                       hypothetical organic
       MW=100.0
C                                       g/cc liquid
       DENSTY=1.0
C                                       Temperature
       T=298.
C                                       Gas pressure, 1 atm in dyne/cm*cm
       PGAS=1.013E+6
C                                       Approximate the Diffusivity
       DIFFUS=0.
C                                       To Autocalculate
       BCE=0.
C
C-----------------------------------------------------------------------
C
```

```
C                              Liquid Molar Volume, cc/mole
      VL=MW/DENSTY
C                              Molecular Volume, cc/molecule (liquid)
      VM=VL/AN
C                              Molecular Diameter, cm
      DIAM=(6.*VM/PI)**TH1
C                              Molecular Surface Area, cm*cm
      SAM=PI*DIAM*DIAM
C                              Concentration (Sat.), molecules/cc
      CS=VP/(BK*T)
C                              0.25 Mean Molecular Velocity, cm/sec
      VELQ=SQRT(RGAS*T/(TWO*PI*MW))
C                              Vapor Pressure, atm
C       VPAT=VP/1.0133E+6
C                              Surface Energy in kT units for monomer
      DIMST=2.*SIGMA*SAM/BK/T/3.
C                              Characteristic Rate Scale, #/cc/sec
      RSCALE=SAM*CS*CS*VELQ
C                              Characteristic Collision Time, sec, sat.
      TB=CS/RSCALE
C                              Characteristic Kelvin Diameter
      DIKELV=4.*SIGMA*VM/(BK*T)
C
C       The following are used only by the cgs condensation rate routines
C
C                Collision Diameter based on BS&L recommendations
C
C                              Collision Diameter of Air (BS&L) [cm]
      CDAIR=3.617E-8
C                              Collision Diameter of Condensible
      CDCON=0.98*DIAM
C                              Collision Diameter (condensible with air)
      COLLDI=(CDCON+CDAIR)/2.
C                              Number Concentration of Air [molecules/cc]
      AIRN=PGAS/BK/T
C                              Air Mean Free Path
      AMFP=1./(SQRT(2.)*AIRN*PI*CDAIR**2)
C                              Condensible M.F.P.
      CMFP=1./(SQRT(1.+MW/29.0)*AIRN*PI*COLLDI**2)
C
C***             Estimate Diffusivity of Monomer in Air if unknown
C
C      IF (BCE.EQ.ZERO.AND.DIFFUS.EQ.ZERO)    DIFFUS=(2./3.)*
C     $      ((RGAS*T/PI)**1.5)*SQRT(0.5/MW+0.5/29.0)/PGAS/(COLLDI**2)/AN
C       The above is taken from illustrative simple theory of BS&L.
C
C       Use Chapman-Enskog theory for better estimate of diffusivity:
C       (re: Jim Davis, AS&T, 1983, as well as eq. 16.4-12 in BS&L)
C
      IF (BCE.EQ.ZERO.AND.DIFFUS.EQ.ZERO)  DIFFUS=(3./8./PI)*
     $      SQRT(PI*RGAS*T*(1./MW+1./29.0)/2.)/((COLLDI**2)*AIRN)
C
C       Note: The Chapman-Enskog DIFFUS is 1.767 of the simple BS&L
```

```
C        illustrative theory prediction (based on 0th order diffusion),
C        assuming a collision integral of unity (in denominator of CE).
C
C        Thus simple BS&L or Fuchs-Sutugin predicts BCE = (1+Z)/6.
C        But Chapmann-Enskog predicts BCE = (1+Z)*3*PI/32.
C
      IF (BCE.EQ.ZERO) BCE=DIFFUS/(4.*VELQ*CMFP)
      IF (DIFFUS.EQ.ZERO)  DIFFUS=4.*VELQ*CMFP*BCE
C
C    Equivalent Monomer Knudsen Number (not based on true MFP)
      DIMKN=6.*DIFFUS/(4.*VELQ*DIAM)
C
C
      RETURN
      END
      SUBROUTINE DBPSET(T,DIMST,DIMKN,TB,CS)

C***********************************************************************
C
C  PURPOSE:     To Set Dimensional and Dimensionless Physical Parameters
C
C  ON ENTRY:
C  T       Temperature (K)
C
C  ON RETURN:
C       DIMST   Dimensionless Surface Tension
C       DIMKN   Dimensionless Monomer Knudsen Number
C       TB      Characteristic Collision Time (Saturated Vapor) [sec]
C       CS      Saturated Vapor Concentration [#/cc]
C       /DS/    all COMMON variables set (dimensional properties)
C
C  COMMENTS:
C       This subroutine must be specialized for each compound.
C
C***********************************************************************

      PARAMETER ( PI = 3.1415927 )
      PARAMETER ( ZERO=0. , ONE=1. , TWO=2. , THREE=3. )
      PARAMETER ( TH1=0.3333333 , TH2=0.6666667 )
C                           Gas Constant, erg/K/mole
      PARAMETER ( RGAS = 8.31433E+7 )
C                           Boltzmann Constant, erg/K/molecule
      PARAMETER ( BK = 1.38054E-16 )
C                           Avogadro's Number, molecules/mole
      PARAMETER ( AN = 6.02252E+23 )
      REAL MW
C
      COMMON /DS/ SIGMA,VP,DIAM,MW,DIFFUS,BCE,RSCALE,DIKELV,CMFP,AMFP,
     $                   DENSTY
C
C-----------------------------------------------------------------------
C
      TEMP = T - 273.16

C  *** The following properties are for DBP


C                                  DBP g/gmole
      MW = 278.35
C
C                                  g/cc liquid
      DENSTY = 1.063 - 0.000826 * TEMP

C                                  Surface Tension
      SIGMA = 35.3 - 0.0863 * TEMP

C                                  ln (vapor pressure, mm Hg)
```

```
      TORRLN = 16.27 - (5099.0 / (T - 109.51))

C                                         Vapor Pressure, dyne/cm*cm
      VP = 1.3332E+3 * EXP (TORRLN)
C                                         Gas pressure, 1 atm in dyne/cm*cm
      PGAS=1.013E+6
C                                         Approximate the Diffusivity
      DIFFUS=0.
C                                         To Autocalculate
      BCE=0.
C
C----------------------------------------------------------------------
C
C                                 Liquid Molar Volume, cc/mole
      VL=MW/DENSTY
C                                 Molecular Volume, cc/molecule (liquid)
      VM=VL/AN
C                                 Molecular Diameter, cm
      DIAM=(6.*VM/PI)**TH1
C                                 Molecular Surface Area, cm*cm
      SAM=PI*DIAM*DIAM
C                                 Concentration (Sat.), molecules/cc
      CS=VP/(BK*T)
C                                 0.25 Mean Molecular Velocity, cm/sec
      VELQ=SQRT(RGAS*T/(TWO*PI*MW))
C                                 Vapor Pressure, atm
C       VPAT=VP/1.0133E+6
C                                 Surface Energy in kT units for monomer
      DIMST=2.*SIGMA*SAM/BK/T/3.
C                                 Characteristic Rate Scale, #/cc/sec
      RSCALE=SAM*CS*CS*VELQ
C                                 Characteristic Collision Time, sec, sat.
      TB=CS/RSCALE
C                                 Characteristic Kelvin Diameter
      DIKELV=4.*SIGMA*VM/(BK*T)
C
C       The following are used only by the cgs condensation rate routines
C
C               Collision Diameter based on BS&L recommendations
C
C                                 Collision Diameter of Air (BS&L) [cm]
      CDAIR=3.617E-8
C                                 Collision Diameter of Condensible
      CDCON=0.98*DIAM
C                                 Collision Diameter (condensible with air)
      COLLDI=(CDCON+CDAIR)/2.
C                                 Number Concentration of Air [molecules/cc]
      AIRN=PGAS/BK/T
C                                 Air Mean Free Path
      AMFP=1./(SQRT(2.)*AIRN*PI*CDAIR**2)
C                                 Condensible M.F.P.
      CMFP=1./(SQRT(1.+MW/29.0)*AIRN*PI*COLLDI**2)
C
```

```
C***            Estimate Diffusivity of Monomer in Air if unknown
C
C      IF (BCE.EQ.ZERO.AND.DIFFUS.EQ.ZERO)    DIFFUS=(2./3.)*
C     $     ((RGAS*T/PI)**1.5)*SQRT(0.5/MW+0.5/29.0)/PGAS/(COLLDI**2)/AN
C       The above is taken from illustrative simple theory of BS&L.
C
C       Use Chapman-Enskog theory for better estimate of diffusivity:
C       (re: Jim Davis, AS&T, 1983, as well as eq. 16.4-12 in BS&L)
C
      IF (BCE.EQ.ZERO.AND.DIFFUS.EQ.ZERO)  DIFFUS=(3./8./PI)*
     $     SQRT(PI*RGAS*T*(1./MW+1./29.0)/2.)/((COLLDI**2)*AIRN)
C
C       Note: The Chapman-Enskog DIFFUS is 1.767 of the simple BS&L
C       illustrative theory prediction (based on 0th order diffusion),
C       assuming a collision integral of unity (in denominator of CE).
C
C       Thus simple BS&L or Fuchs-Sutugin predicts BCE = (1+Z)/6.
C       But Chapmann-Enskog predicts BCE = (1+Z)*3*PI/32.
C
      IF (BCE.EQ.ZERO) BCE=DIFFUS/(4.*VELQ*CMFP)
      IF (DIFFUS.EQ.ZERO)  DIFFUS=4.*VELQ*CMFP*BCE
C
C    Dimensionless Diffusivity (Kn not based on true MFP)
      DIMKN=6.*DIFFUS/(4.*VELQ*DIAM)
C
C
      RETURN
      END
```

```
C=====================================================================
C
C
C  RSET2 allows the user to enter a selected set of runs
C  All the properties passed may be set as a function of
C   NCASE.  Set NCASE=-1 to quit.
C
      SUBROUTINE RSET2(NCASE,DIMSOR,TRES)
C
C************************************************************************
C
C     The following subroutine will experiment with initial-
C  saturation-ratio-driven homogeneous nucleation experiments
C       with initial aerosol similar to those performed by Kikuo
C       Okuyama in late 1985.                          -- DRW

      LOGICAL INIT,TRUTH
      CHARACTER*1 ASK
      LOGICAL DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      LOGICAL PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /INIT/ SR0,DIMNO,DIMD0,CINIT,DPI,TEMP,CSAT
      COMMON /DFLAGS/ DOKELV, USEBCE, SCALES, HIGHJ, LOTHE, DODEPO
      COMMON /AFLAGS/ PRESET, NODIM, UTEMP, USOR, UHOURS, NOSHO
      COMMON /SNM2/ DIM4(4),GS,FIN1,FIN2,FACTJ
      COMMON /CASE/ NCA,RNJ0

C******************************************************************

      DATA NMAX / 40 /

      UTEMP = .TRUE.
      DIMSOR = .0
      USOR = .FALSE.
      SCALES = .FALSE.
      UHOURS = .FALSE.
      DODEPO = .FALSE.
      LOTHE = .FALSE.
C     HIGHJ = .TRUE.
C     FACTJ = 1.E8
      TRES = 0.05
      FIN1 = 1.0
      FIN2 = 1.0
      SR0 = 250.
      TEMP = 50.
      DPIM = 0.1
      NOSHO = .FALSE.
      IF (NCASE.EQ.1) THEN
         CINIT = 0.
         WRITE(*,20) DPIM
 20      FORMAT(' ENTER Dp (um) for initial aerosol [',F7.4,'] : ',\)
         READ(*,30)  DUMMY
 30      FORMAT(F9.0)
```

```
   IF (DUMMY.GT.O.) DPIM = DUMMY
   DPI = 1.E-4 * DPIM
ELSEIF (NCASE.GT.NMAX) THEN
   NCASE = -1
ELSE
   CINIT = CSAT * RNJO * 10.**(NCASE/5.-4.4)
ENDIF
RETURN
END
```

```
      FUNCTION DEPSIT(DIMD,AKN)
C
C***********************************************************************
C
C  PURPOSE:
C       To Calculate the Deposition Coefficients Due To
C        Gravity, Diffusion, and Electrostatics.
C        The Coefficient is Given For the Overall Container.
C
C  ON ENTRY:
C       DIMD            Dimensionless particle diameter
C       AKN             Knudsen number of above
C
C       /DEPOST/   ELCFLD      Electric field, [V/cm]
C       //         KE          Turbulence parameter [/sec]
C       //         DMON        Monomer diameter [cm]
C       //         DVESSL      Vessel diameter [cm]
C       //         RHO         Particle density [g/cc]
C       //         DIFF        Particle diffusivity [cm**2/sec]
C       //         TGAS        Gas Temperature [K]
C
C  ON RETURN:
C       DEPSIT          Deposition Coefficient [/sec]
C
C  COMMENTS:
C       Based on work by Pete McMurry and, earlier, Jim Crump.
C
C***********************************************************************
C
      REAL*4  KE
      REAL*4  PI
C                       Fraction of particles with charge i
      REAL F(10), F0

      PARAMETER ( PI = 3.1415927 )

C                       Boltzmann Constant, erg/K/molecule
      PARAMETER ( BK = 1.38054E-16 )

C                       Elementary Charge, coulombs
      PARAMETER ( EC = 1.602E-19 )

C                       Elementary Charge in esu, E_IN_ESU
      PARAMETER ( EINESU=4.80286E-10 )

      COMMON /DEPOST/ ELCFLD,KE,DMON,DVESSL,RHO,DIFF,ICHRG,TGAS

C                       Viscosity of air, [g/cmsec]
C                       Should add temperature dependence.
      VISC=1.84E-4

C                       Back into dimensional form
```

```
      DP=DIMD*DMON
C                         Cunningham correction
      CUNN=1.+AKN*(1.257+0.4*EXP(-1.1/AKN))

C  For electrostatic deposition:

C                                   Particle diffusivity by Stokes-Einstein
      DIFF = (BK*TGAS*CUNN)/(3.*PI*VISC*DP)

      SQRKED=SQRT(DIFF*KE)

C                         Terminal Velocity
      VTERM=RHO*DP**2*CUNN*980./(18.*VISC)
      Z=0.5*PI*VTERM/SQRKED

C                         Assuming singly-charged particles
      IF (ICHRG.EQ.1) THEN
C                         Number of charges per particle
        CHNUM=1.
C                         Electrostatic Velocity
        VEBAR=NCHG*EC*CUNN*ELCFLD*1.E7/(3.*PI*VISC*DP)
        Y=0.5*PI*VEBAR/SQRKED
C                         Dep. coeff. for singly charged
        DEPSIT=DEP(SQRKED,DVESSL,Y,Z)
        RETURN

      ELSE

C Using a Boltzmann charge distribution only if Dp is > 0.03 um.
C  If Dp is less than this, use an approximation by Knutson to a more
C  complicated formula by Gentry for the number of singly
C  charged particles versus the number of neutral particles.


       IF (DP.LE.3.E-6) THEN
C                         Fraction of singly charged
         FSINGL = 1./(2.+ (1.E-5/DP)**1.544)
         CHNUM=1.
          VEBAR=ABS(CHNUM*EC*CUNN*ELCFLD/(3*PI*VISC*DP))
          Y=0.5*PI*VEBAR/SQRKED
C                         Dep. coeff. for singly charged
         DEPSNG=DEP(SQRKED,DVESSL,Y,Z)

C                         Fraction of neutral
         FNUTRL = 1.- (2.*FSINGL)
C                         Electrostatic contribution is 0
         Y=0.
C                         Dep. coeff. for neutral
         DEPNTR=DEP(SQRKED,DVESSL,Y,Z)

C                         Average dep. coeff.
         DEPSIT = FNUTRL*DEPNTR + 2.*FSINGL*DEPSNG
         RETURN
```

```
C                          Regular Boltzmann distribution
       ELSE

C                          .5 puts Dp into radius
        SIGMA=SQRT(0.5*DP*BK*TGAS/EINESU**2)
        SUM=0.

        DO 200 I=1,10
C
C                          Fraction of particle with charge i
         F(I)=EXP(-I**2/(2.*SIGMA**2))/(SQRT(2.*PI)*SIGMA)
         CHNUM=I

          VEBAR=ABS(CHNUM*EC*CUNN*ELCFLD/(3*PI*VISC*DP))
          Y=0.5*PI*VEBAR/SQRKED
C
C                          Dep. coeff. for each # of charges
          DEPOFI=DEP(SQRKED,DVESSL,Y,Z)

C                          Sum of Dep_of_i for i=-10,10
           SUM=SUM+2.*DEPOFI*F(I)

 200     CONTINUE
C
C                  Now consider uncharged particles
C
         FO = 1. / (SQRT(2.*PI)*SIGMA)
         Y=0.
         DEPOFI = DEP(SQRKED,DVESSL,Y,Z)
         SUM = SUM + DEPOFI*F(I)
C

         DEPSIT=SUM
         RETURN

       END IF
C                  IF on particle size (within Boltzmann assumption)

      END IF
C        IF on singly charged versus Boltzman charged

      END

C***********************************************************************


      FUNCTION DEP(SQRKED,DIAMET,Y,Z)

      PARAMETER (PI=3.1415927)

         IF (Z.GE.Y) THEN
C
```

```
C                 Pete McMurry's expression
C
            DEP=(6*SQRKED/(PI*DIAMET*Z))*(0.5*(Z+Y)**2+(Z+Y)
     %       *DEBYE1(Z+Y)+(Z-Y)*DEBYE1(Z-Y))
C
          ELSE
C
C                 rederived for y>z
C
            DEP=(6*SQRKED/(PI*DIAMET*Z))*(2.*(Z+Y)+(Z+Y)*DEBYE1(Z+Y)
     %       +(Z-Y)*DEBYE1(Y-Z))
C
          END IF
      RETURN
      END

C-----------------------------------------------------------------

      FUNCTION DEBYE1(X)
      PARAMETER ( NTABLE = 43 )
      DIMENSION X1(NTABLE),D1(NTABLE)

C     Compute Debye Function, Order 1, By Interpolating a Look-up Table
C     The linear interpolation should be accurate to nearly 4 sig figs

      DATA X1 / 0.0 , 0.1 , 0.2 , 0.3 , 0.4 , 0.5 , 0.6 , 0.7 ,
     $ 0.8 , 0.9 , 1.0 , 1.1 , 1.2 , 1.3 , 1.4 , 1.6 , 1.8 ,
     $ 2.0 , 2.2 , 2.4 , 2.6 , 2.8 , 3.0 , 3.2 , 3.4 , 3.6 ,
     $ 3.8 , 4.0 , 4.2 , 4.4 , 4.6 , 4.8 , 5.0 , 5.5 , 6.0 ,
     $ 6.5 , 7.0 , 7.5 , 8.0 , 8.5 , 9.0 , 9.5 , 10.0 /
      DATA D1 / 1.00,0.9753,0.9511,0.9275,0.9044,0.8819,0.8600,0.8385,
     $ 0.8177,0.7973,0.7775,0.7582,0.7394,0.7212,0.7034,0.6694,0.6372,
     $ 0.6069,0.5784,0.5516,0.5264,0.5027,0.4804,0.4596,0.4400,0.4216,
     $ 0.4043,0.3881,0.3730,0.3587,0.3453,0.3327,0.3209,0.2942,0.2713,
     $ 0.2513,0.2339,0.2187,0.2052,0.1933,0.1826,0.1731,0.1644 /
C
      IF (X.LT.X1(1)) THEN
        WRITE(*,55)
55      FORMAT(/' BAD ARGUMENT TO DEBYE1'/)
        DEBYE1=1.
        RETURN
      END IF
C
C       Find place in table, I is first I such that X1(I) larger than X
C
      I=2
 100  IF (X.LT.X1(I)) GOTO 200
        I=I+1
        IF (I.GT.NTABLE) GOTO 500
      GOTO 100
C
C       Interpolation between tabulated values
C
```

```
 200  DEBYE1=D1(I)+(X-X1(I))*(D1(I)-D1(I-1))/(X1(I)-X1(I-1))
      RETURN
C
C       For Large X, use asymptotic approximation:
C
 500  DEBYE1=D1(NTABLE)*X1(NTABLE)/X
      RETURN
      END
```

```
C=====================================================================

      SUBROUTINE PEDERV(N,T,Y,PD,NO)

C This is presently intended to be a dummy subroutine in this application
C Used only in EPISODE versions of MAEROS ; not adequate if MF=11 or 21

      WRITE(*,10) T
   10 FORMAT(/5X,' Error -- PEDERV was called at time ',1PE10.2/)
      WRITE(*,20)
   20 FORMAT(' Hence MITER of MF was set equal to one'/)
      STOP 'STOP on bad MF to DRIVES for Dummy PEDERV'
      END
```

```
$DEBUG
      SUBROUTINE DRIVE (N, TO, HO, YO, TOUT, EPS, IERROR, MF, INDEX)
C
C*************************************************************************
C
C  PURPOSE:
C  To Solve a System of Stiff ODEs, with custom modifications to
C   handle a non-negativity constraint and to keep error limited
C   where neither simple relative nor absolute error bounds
C   are appropriate.
C
C  ON ENTRY:
C  See original documentation below.
C
C  ON RETURN:
C  See original documentation below.
C
C  COMMENTS:
C  This is the November 1982 Modification (called EPIS)
C   by Dale Warren (Caltech) of the EPISODE program.
C  Adapted in February 1986 for the IBM AT running
C   MICROSOFT FORTRAN-77 v3.20.
C  The ability to convert to DOUBLE PRECISION has been
C   removed (for conciseness and readability).
C
C*************************************************************************
C          THE JUNE 24, 1975  VERSION OF
C EPISODE..  EXPERIMENTAL PACKAGE FOR INTEGRATION OF
C SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS,
C    DY/DT = F(Y,T),  Y = (Y(1),Y(2),...,Y(N)) TRANSPOSE,
C GIVEN THE INITIAL VALUE OF Y.
C THIS CODE IS FOR THE IBM 370/195 AT ARGONNE NATIONAL LABORATORY
C AND IS A MODIFICATION OF EARLIER VERSIONS BY G.D.BYRNE
C AND A.C.HINDMARSH.
C                    REFERENCES
C 1.  G. D. BYRNE AND A. C. HINDMARSH, A POLYALGORITHM FOR THE
C        NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS,

C        UCRL-75652, LAWRENCE LIVERMORE LABORATORY, P. O. BOX 808,
C        LIVERMORE, CA 94550, APRIL 1974. ALSO IN ACM TRANSACTIONS
C        ON MATHEMATICAL SOFTWARE, 1 (1975), PP. 71-96.
C
C 2.  A. C. HINDMARSH AND G. D. BYRNE, EPISODE.. AN EXPERIMENTAL
C        PACKAGE FOR THE INTEGRATION OF SYSTEMS OF ORDINARY
C        DIFFERENTIAL EQUATIONS, UCID-30112, L.L.L., MAY, 1975.
C
C 3.  A. C. HINDMARSH, GEAR.. ORDINARY DIFFERENTIAL EQUATION
C        SYSTEM SOLVER, UCID-30001, REV. 3, L.L.L., DECEMBER, 1974.
C
C-------------------------------------------------------------------------
C DRIVE IS A DRIVER SUBROUTINE FOR THE EPISODE PACKAGE.
C DRIVE IS TO BE CALLED ONCE FOR EACH OUTPUT VALUE OF T.
```

```
C IT THEN MAKES REPEATED CALLS TO THE CORE INTEGRATOR
C SUBROUTINE, TSTEP.
C
C THE INPUT PARAMETERS ARE AS FOLLOWS.
C   N      =  THE NUMBER OF DIFFERENTIAL EQUATIONS (USED ONLY ON
C               FIRST CALL, UNLESS INDEX = -1).  N MUST NEVER BE
C               INCREASED DURING A GIVEN PROBLEM.
C   TO     =  THE INITIAL VALUE OF T, THE INDEPENDENT VARIABLE
C               (USED FOR INPUT ONLY ON FIRST CALL).
C   HO     =  THE STEP SIZE IN T (USED FOR INPUT ONLY ON THE
C               FIRST CALL, UNLESS INDEX = 3 ON INPUT).  WHEN
C               INDEX = 3, HO IS THE MAXIMUM ABSOLUTE VALUE OF
C               THE STEP SIZE TO BE USED.
C   YO     =  A VECTOR OF LENGTH N CONTAINING THE INITIAL VALUES OF
C               Y (USED FOR INPUT ONLY ON FIRST CALL).
C   TOUT   =  THE VALUE OF T AT WHICH OUTPUT IS DESIRED NEXT.
C               INTEGRATION WILL NORMALLY GO BEYOND TOUT AND
C               INTERPOLATE TO T = TOUT.  (USED ONLY FOR INPUT.)
C   EPS    =  THE RELATIVE ERROR BOUND (USED ONLY ON FIRST CALL,
C               UNLESS INDEX = -1).  THIS BOUND IS USED AS FOLLOWS.
C               LET R(I) DENOTE THE ESTIMATED RELATIVE LOCAL ERROR
C               IN Y(I), I.E. THE ERROR RELATIVE TO YMAX(I), AS
C               MEASURED PER STEP (OF SIZE H) OR PER SS UNITS OF T.
C               THEN EPS IS A BOUND ON THE ROOT-MEAN-SQUARE NORM
C               OF THE VECTOR R, I.E.
C                       N
C               SQRT ( SUM ( R(I)**2 )/N ) .LT. EPS.
C                      I=1
C               THE VECTOR YMAX IS COMPUTED IN DRIVE AS DESCRIBED
C               UNDER IERROR BELOW.
C               IF ERROR CONTROL PER SS UNITS OF T IS DESIRED, SET SS
C               TO A POSITIVE NUMBER AFTER STATEMENT 10 (WHERE IT IS
C               NOW SET TO ZERO) AND UPDATE IT AFTER STATEMENT 60.
C               SEE ALSO THE COMMENTS ON SS AND YMAX BELOW.
C  IERROR =  THE ERROR FLAG WITH VALUES AND MEANINGS AS FOLLOW.
C         1     ABSOLUTE ERROR IS CONTROLLED.  YMAX(I) = 1.0.
C         2     ERROR RELATIVE TO ABS(Y) IS CONTROLLED.  IF Y(I) = 0.0
C               A DIVIDE ERROR WILL Not OCCUR.  YMAX(I) = ABS(Y(I)).
C         3     ERROR RELATIVE TO THE LARGEST VALUE OF ABS(Y(I)) SEEN
C               SO FAR IS CONTROLLED.  IF THE INITIAL VALUE OF Y(I) IS
C               0.0, THEN YMAX(I) IS SET TO 1.0 INITIALLY AND REMAINS
C               AT LEAST 1.0.
C         4     SAME AS 2 EXCEPT IF Y(I) INITIALLY <YMIN, YMAX(I)=YMIN
C         5     SAME AS 3 EXCEPT IF Y(I) CURRENTLY <YMIN, YMAX(I)=YMIN
C         6     SAME AS 4 EXCEPT IF Y(I) < 0., Error Criteria Not Met
C         7     SAME AS 5 EXCEPT IF Y(I) < 0., Error Criteria Not Met
C         8     SAME AS 4 EXCEPT IF Y(I) < -YMIN, Error Criteria Not Met
C         9     SAME AS 5 EXCEPT IF Y(I) < -YMIN, Error Criteria Not Met
C            Note: For 6-9, Special Modification so Y(N)<0. rejected
C               4 & 5 were added for problems when
C                IERROR=2 fails because of divide by zero and
C                IERROR=3 scales poorly to ONE       -DRW
C          Note 4 & 5 require user to set YMIN reasonably in DRIVES
```

```
C   MF    =  THE METHOD FLAG (USED ONLY ON FIRST CALL, UNLESS
C              INDEX = -1).  ALLOWED VALUES ARE 10, 11, 12, 13,
C              20, 21, 22, 23.  MF IS AN INTEGER WITH TWO DECIMAL
C              DIGITS, METH AND MITER (MF = 10*METH + MITER).  (MF
C              CAN BE THOUGHT OF AS THE ORDERED PAIR (METH,MITER).)
C              METH IS THE BASIC METHOD INDICATOR.
C                METH = 1 INDICATES VARIABLE-STEP SIZE, VARIABLE-
C                         ORDER ADAMS METHOD, SUITABLE FOR NON-
C                         STIFF PROBLEMS.
C                METH = 2 INDICATES VARIABLE-STEP SIZE, VARIABLE-
C                         ORDER BACKWARD DIFFERENTIATION METHOD,
C                         SUITABLE FOR STIFF PROBLEMS.
C              MITER INDICATES THE METHOD OF ITERATIVE CORRECTION
C                (NONLINEAR SYSTEM SOLUTION).
C                MITER = 0 INDICATES FUNCTIONAL ITERATION (NO
C                          PARTIAL DERIVATIVES NEEDED).
C                MITER = 1 INDICATES A CHORD OR SEMI-STATIONARY
C                          NEWTON METHOD WITH CLOSED FORM (EXACT)
C                          JACOBIAN, WHICH IS COMPUTED IN THE
C                          USER SUPPLIED SUBROUTINE
C                          PEDERV(N,T,Y,PD,NO) DESCRIBED BELOW.
C                MITER = 2 INDICATES A CHORD  OR SEMI-STATIONARY
C                          NEWTON METHOD WITH AN INTERNALLY
C                          COMPUTED FINITE DIFFERENCE APPROXIMATION
C                          TO THE JACOBIAN.
C                MITER = 3 INDICATES A CHORD OR SEMI-STATIONARY
C                          NEWTON METHOD WITH AN INTERNALLY
C                          COMPUTED DIAGONAL MATRIX APPROXIMATION
C                          TO THE JACOBIAN, BASED ON A DIRECTIONAL
C                          DERIVATIVE.
C   INDEX =  INTEGER USED ON INPUT TO INDICATE TYPE OF CALL,
C              WITH THE FOLLOWING VALUES AND MEANINGS..
C        1     THIS IS THE FIRST CALL FOR THIS PROBLEM.
C        0     THIS IS NOT THE FIRST CALL FOR THIS PROBLEM,
C              AND INTEGRATION IS TO CONTINUE.
C       -1     THIS IS NOT THE FIRST CALL FOR THE PROBLEM,
C              AND THE USER HAS RESET N, EPS, AND/OR MF.
C        2     SAME AS 0 EXCEPT THAT TOUT IS TO BE HIT
C              EXACTLY (NO INTERPOLATION IS DONE).
C              ASSUMES TOUT .GE. THE CURRENT T.
C        3     SAME AS 0 EXCEPT CONTROL RETURNS TO CALLING
C              PROGRAM AFTER ONE STEP.  TOUT IS IGNORED.
C        7     THIS IS NOT THE FIRST CALL, BUT THE Y ARRAY
C              HAS CHANGED SLIGHTLY, SO THE DERIVATIVES
C              MUST BE RECOMPUTED (NEW by DRW)
C            SINCE THE NORMAL OUTPUT VALUE OF INDEX IS 0,
C            IT NEED NOT BE RESET FOR NORMAL CONTINUATION.
C            SINCE THE NORMAL OUTPUT VALUE OF INDEX IS 0,
C            IT NEED NOT BE RESET FOR NORMAL CONTINUATION.
C
C AFTER THE INITIAL CALL, IF A NORMAL RETURN OCCURRED AND A NORMAL
C CONTINUATION IS DESIRED, SIMPLY RESET TOUT AND CALL AGAIN.
C ALL OTHER PARAMETERS WILL BE READY FOR THE NEXT CALL.
```

```
C A CHANGE OF PARAMETERS WITH INDEX = -1 CAN BE MADE AFTER
C EITHER A SUCCESSFUL OR AN UNSUCCESSFUL RETURN.
C
C THE OUTPUT PARAMETERS ARE AS FOLLOWS.
C   TO     =  THE OUTPUT VALUE OF T.  IF INTEGRATION WAS SUCCESSFUL,
C             TO = TOUT.  OTHERWISE, TO IS THE LAST VALUE OF T
C             REACHED SUCCESSFULLY.
C   HO     =  THE STEP SIZE H USED LAST, WHETHER SUCCESSFULLY OR NOT.
C   YO     =  THE COMPUTED VALUES OF Y AT T = TO.
C   INDEX =  INTEGER USED ON OUTPUT TO INDICATE RESULTS,
C               WITH THE FOLLOWING VALUES AND MEANINGS..
C          0    INTEGRATION WAS COMPLETED TO TOUT OR BEYOND.
C         -1    THE INTEGRATION WAS HALTED AFTER FAILING TO PASS THE
C               ERROR TEST EVEN AFTER REDUCING H BY A FACTOR OF
C               1.E10 FROM ITS INITIAL VALUE.
C         -2    AFTER SOME INITIAL SUCCESS, THE INTEGRATION WAS
C               HALTED EITHER BY REPEATED ERROR TEST FAILURES OR
C               BY A TEST ON EPS.  POSSIBLY TOO MUCH ACCURACY HAS
C               BEEN REQUESTED, OR A BAD CHOICE OF MF WAS MADE.
C         -3    THE INTEGRATION WAS HALTED AFTER FAILING TO ACHIEVE
C               CORRECTOR CONVERGENCE EVEN AFTER REDUCING H BY A
C               FACTOR OF 1.E10 FROM ITS INITIAL VALUE.
C         -4    IMMEDIATE HALT BECAUSE OF ILLEGAL VALUES OF INPUT
C               PARAMETERS.  SEE PRINTED MESSAGE.
C         -5    INDEX WAS -1 ON INPUT, BUT THE DESIRED CHANGES OF
C               PARAMETERS WERE NOT IMPLEMENTED BECAUSE TOUT
C               WAS NOT BEYOND T.  INTERPOLATION TO T = TOUT WAS
C               PERFORMED AS ON A NORMAL RETURN.  TO CONTINUE,
C               SIMPLY CALL AGAIN WITH INDEX = -1 AND A NEW TOUT.
C         -6    INDEX WAS 2 ON INPUT, BUT TOUT WAS NOT BEYOND T.
C               NO ACTION WAS TAKEN.
C   -7    INTEGRATION SUSPENDED BECAUSE A Y(I)<0 FOUND,
C               WITH NRMIN<=I<=NRMAX, AND IERROR OF 6 OR 7
C               HAD PROSCRIBED AGAINST NEGATIVE VALUES --DRW
C
C IN ADDITION TO DRIVE, THE FOLLOWING SUBROUTINES ARE USED BY AND
C PROVIDED IN THIS PACKAGE:
C   INTERP(TOUT,Y,NO,YO)  INTERPOLATES TO GIVE OUTPUT VALUES AT
C                         T = TOUT  BY USING DATA IN THE Y ARRAY.
C   TSTEP(Y,NO)  IS THE CORE INTEGRATION SUBROUTINE, WHICH INTEGRATES
C                OVER A SINGLE STEP AND DOES ASSOCIATED ERROR
C                CONTROL.
C   COSET  SETS COEFFICIENTS FOR USE IN TSTEP.
C   ADJUST(Y,NO)  ADJUSTS THE HISTORY ARRAY Y ON REDUCTION OF ORDER.
C   PSET(Y,NO,CON,MITER,IER)  COMPUTES AND PROCESSES THE JACOBIAN
C                             MATRIX, J = DF/DY.
C   DEC(N,NO,A,IP,IER)  PERFORMS THE LU DECOMPOSITION OF A MATRIX.
C   SOL(N,NO,A,B,IP)  SOLVES A LINEAR SYSTEM A*X = B, AFTER DEC
C                       HAS BEEN CALLED FOR THE MATRIX A.
C NOTE:  PSET, DEC, AND SOL ARE CALLED IF AND ONLY IF MITER = 1
C        OR MITER = 2.
C
C THE USER MUST FURNISH THE FOLLOWING SUBROUTINES:
```

```
C    DIFFUN(N,T,Y,YDOT)  COMPUTES THE FUNCTION  YDOT = F(Y,T),
C                        THE RIGHT HAND SIDE OF THE ORDINARY
C                        DIFFERENTIAL EQUATION SYSTEM, WHERE Y
C                        AND YDOT ARE VECTORS OF LENGTH N.
C    PEDERV(N,T,Y,PD,NO)  COMPUTES THE N BY N JACOBIAN MATRIX OF
C                         PARTIAL DERIVATIVES AND STORES IT IN PD AS
C                         AN NO BY NO ARRAY.   PD(I,J) IS TO BE SET
C                         TO THE PARTIAL DERIVATIVE OF YDOT(I) WITH
C                         RESPECT TO Y(J).  PEDERV IS CALLED IF AND
C                         ONLY IF MITER = 1. FOR OTHER VALUES OF
C                         MITER, PEDERV CAN BE A DUMMY SUBROUTINE.
C
C CAUTION:  AT THE PRESENT TIME THE MAXIMUM NUMBER OF DIFFERENTIAL
C           EQUATIONS, WHICH CAN BE SOLVED BY EPISODE, IS 20.  TO
C           CHANGE THIS NUMBER TO A NEW VALUE, SAY NMAX, CHANGE
C           Y(20,13) TO Y(NMAX,13), YMAX(20) TO YMAX(NMAX),
C           ERROR(20) TO ERROR(NMAX), SAVE1(20) TO SAVE1(NMAX),
C           SAVE2(20) TO SAVE2(NMAX), PW(400) TO PW(NMAX*NMAX),
C           AND IPIV(20) TO IPIV(NMAX) IN THE COMMON AND DIMENSION
C           STATEMENTS BELOW.  ALSO CHANGE THE ARGUMENT IN THE
C           IF...GO TO 440 STATEMENT (AFTER THE COMMON STATEMENTS)
C           FROM 20 TO NMAX.  NO OTHER CHANGES NEED TO BE MADE TO
C           ANY OTHER SUBROUTINE IN THIS PACKAGE WHEN THE MAXIMUM
C           NUMBER OF EQUATIONS IS CHANGED.  ELSEWHERE, THE COLUMN
C           LENGTH OF THE Y ARRAY IS NO INSTEAD OF 20.  THE ROW
C           LENGTH OF Y CAN BE REDUCED FROM 13 TO 6 IF METH = 2.
C           THE ARRAY IPIV IS USED IF AND ONLY IF MITER = 1 OR
C           MITER = 2.  THE SIZE OF THE PW ARRAY CAN BE REDUCED
C           TO 1 IF MITER = 0 OR TO N IF MITER = 3.
C
C THE COMMON BLOCK EPCOM9 CAN BE ACCESSED EXTERNALLY BY THE USER,
C IF HE DESIRES.  IT CONTAINS THE STEP SIZE LAST USED SUCCESSFULLY
C (HUSED), THE ORDER LAST USED SUCCESSFULLY (NQUSED), THE
C NUMBER OF STEPS TAKEN SO FAR (NSTEP), THE NUMBER OF FUNCTION
C EVALUATIONS (DIFFUN CALLS) SO FAR (NFE), AND THE NUMBER OF
C JACOBIAN EVALUATIONS SO FAR (NJE).
C
C IN A DATA STATEMENT BELOW, LOUT IS SET TO THE LOGICAL UNIT NUMBER
C FOR THE OUTPUT OF MESSAGES DURING INTEGRATION.  CURRENTLY, LOUT
C = 3.
C-----------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE DRIVE.
C
C                         Set Maximum number of dif. eqs.
      PARAMETER ( NMAX =120 )
      PARAMETER ( NMAXSQ = NMAX*NMAX )
C*
C  All the Explicit Variable Type Definitions are Unnecessary
C  Simply insert the following card in the each module
C       IMPLICIT REAL*#(A-H,O-Z) , INTEGER(I-N)
C  Where # is 8 for Double Precision and 4 for Single Precision
C*
      INTEGER IERROR, INDEX, MF, N
      INTEGER IPIV, JSTART, KFLAG, MFC, NC, NFE, NJE,
     1        NQUSED, NSQ, NSTEP
      INTEGER I, KGO, NHCUT, NO
      INTEGER LOUT
      INTEGER NFLAG
      REAL EPS, HO, TOUT, TO, YO
      REAL EPSC, EPSJ, ERROR, HMAX, H, HMIN, HUSED,
     1     PW, SAVE1, SAVE2, SS, T, UROUND, YMAX
      REAL AYI, D, TOP, Y
      REAL HCUT
      REAL FOUR, HUNDRD, ONE, TEN, ZERO
      REAL*4 YMIN,YCUT
      DIMENSION Y(NMAX,13)
      DIMENSION YO(N)
C
      COMMON /EPCOM1/T,H,HMIN,HMAX,EPSC,SS,UROUND,NC,MFC,KFLAG,JSTART
      COMMON /EPCOM2/ YMAX(NMAX)
      COMMON /EPCOM3/ ERROR(NMAX)
      COMMON /EPCOM4/ SAVE1(NMAX)
      COMMON /EPCOM5/ SAVE2(NMAX)
      COMMON /EPCOM6/ PW(NMAXSQ)
      COMMON /EPCOM7/ IPIV(NMAX)
      COMMON /EPCOM8/ EPSJ,NSQ
      COMMON /EPCOM9/ HUSED,NQUSED,NSTEP,NFE,NJE
C                                    For # of Evaluations
      COMMON /EPC099/ NCSTEP,NCFE,NCJE
C                                    Set by calling prog - DRW
      COMMON /EPCOMR/ NRMIN,NRMAX
C                                    Set by calling prog - DRW
      COMMON /EPCOMY/ YMIN,HMAXMX
C
C                             Messages to Unit # 3, or FOR003.DAT
      DATA LOUT /11/
      DATA HCUT /0.1E0/
      DATA FOUR /4.0E0/, HUNDRD /1.0E2/, ONE  /1.0E0/,
     1     TEN  /1.0E1/, ZERO   /0.0E0/
C                   Convenient for Nucleation Tests - DRW
C     DATA YMIN /1.0E-17/
C                             Normal Continuation
      IF (INDEX .EQ. 0) GOTO 20
C                             Continue & Hit Exactly
      IF (INDEX .EQ. 2) GOTO 25
C                             Integration Mode Reset
      IF (INDEX .EQ. -1) GOTO 30
C                             Single Step Integration
      IF (INDEX .EQ. 3) GOTO 40
C                             NEW -- Continue with Y modified
      IF (INDEX .EQ. 7) GOTO 27
C                             Bad Input; 1 is First Call
      IF (INDEX .NE. 1) GOTO 430
C
      IF (EPS .LE. ZERO) GOTO 400
      IF (N .LE. 0) GOTO 410
      IF (N .GT. NMAX) GOTO 440
      IF ((TO-TOUT)*HO .GE. ZERO) GOTO 420
      WRITE(LOUT,999) TO,HO,TOUT
  999 FORMAT(1H ,2X,'TO=',E12.5,2X,'HO=',E12.5,2X,'TOUT=',E12.5)
C-----------------------------------------------------------------------
C IF INITIAL VALUES FOR YMAX OTHER THAN THOSE BELOW ARE DESIRED,
C THEY SHOULD BE SET HERE.  ALL YMAX(I) MUST BE POSITIVE.  IF
C VALUES FOR HMIN OR HMAX, THE BOUNDS ON THE ABSOLUTE VALUE OF H,
C OTHER THAN THOSE BELOW, ARE DESIRED, THEY ALSO SHOULD BE SET HERE.
C IF ERROR PER SS UNITS OF T IS TO BE CONTROLLED, SS SHOULD BE SET
C TO A POSITIVE VALUE BELOW.  ERROR PER UNIT STEP IS CONTROLLED
C WHEN SS = 1.  THE DEFAULT VALUE FOR SS IS 0 AND YIELDS CONTROL
C OF ERROR PER STEP.
C-----------------------------------------------------------------------
C SET UROUND, THE MACHINE ROUNDOFF CONSTANT, HERE.
C USE STATEMENT BELOW FOR SHORT PRECISION ON IBM 360 OR 370.
C     UROUND = 9.53674E-7
C USE STATEMENT BELOW FOR SINGLE PRECISION ON CDC 7600 OR 6600.
C     UROUND = 7.105427406E-15
C USE STATEMENT BELOW FOR LONG PRECISION ON IBM 360 OR 370.
C-----------------------------------------------------------------------
C                            Set for VAX Single Precision
      UROUND = 5.960E-8
      IF (IERROR.LE.5) GOTO 3
C                            Default check for negative Y(I)
      IF (NRMIN.LE.0) NRMIN=1
C                            or, for all values of I
C      IF (NRMAX.LE.0) NRMAX=N
C                            Special for MAEROS with Vapor
      IF (NRMAX.LE.0) NRMAX=N-1
C
C
  3   DO 10 I = 1,N
C
   GOTO (5, 6, 7, 8, 8, 8, 8, 8, 8), IERROR
C IERROR  =   1, 2, 3, 4, 5, 6, 7, 8, 9 ------Six Extra by DRW----------
C
C                                      For ABSOLUTE Error
  5     YMAX(I) = ONE
   GOTO 10
  6     YMAX(I) = ABS(YO(I))
C                                      To Avoid Automatic /O
        IF (YMAX(I).EQ.ZERO) YMAX(I)=YMIN
        GOTO 10
  7     YMAX(I) = ABS(YO(I))
C                                      For SEMI-RELATIVE to 1.
   IF (YMAX(I) .EQ. ZERO) YMAX(I) = ONE
        GOTO 10
C                                      For RELATIVE Error
  8     YMAX(I) = ABS(YO(I))
C     WRITE(LOUT,998) I,YMAX(I),ABS(YO(I)),YMIN
C 998 FORMAT(1H ,2X,'I=',I2,2X,'YMAX=',E12.5,2X,'ABY=',E12.5,2X,
C    $'YMIN=',E12.5)
```

```
        IF (YMAX(I) .LT. YMIN) YMAX(I) = YMIN
 10     Y(I,1) = YO(I)
      NC = N
      T = TO
      H = HO
      WRITE(LOUT,997) NC,T,H
  997 FORMAT(1H ,2X,'NC=',I2,2X,'T=',E12.5,2X,'H=',E12.5)
      IF ((T+H) .EQ. T) WRITE(LOUT,15) T
CO 15   FORMAT(/46H---  MESSAGE FROM SUBROUTINE DRIVE IN EPISODE,,
CO     1        24H THE O.D.E. SOLVER.  ---/22H WARNING.. T + H = T =,
CO     2        E18.8,18H IN THE NEXT STEP./)
 15   FORMAT(' WARNING... T + H = T =',1PE16.7,' IN THE NEXT STEP.')
      HMIN = ABS(HO)
      HMAX = ABS(TO - TOUT)*TEN
      HMAX = AMIN1(HMAX,HMAXMX)
      EPSC = EPS
      MFC = MF
      JSTART = 0
      SS = ZERO
      NO = N
      NSQ = NO*NO
      EPSJ = SQRT(UROUND)
      NHCUT = 0
      YCUT= ZERO
      IF (IERROR.GE.8) YCUT=-YMIN
      GOTO 50
C TOP IS THE PREVIOUS OUTPUT VALUE OF TO FOR USE IN HMAX. --------------
  20  HMAX = ABS(TOUT - TOP)*TEN
      HMAX = AMIN1(HMAX,HMAXMX)
      GOTO 80
  25  HMAX = ABS(TOUT - TOP)*TEN
      HMAX = AMIN1(HMAX,HMAXMX)
C
       IF ((T-TOUT)*H .GE. ZERO) GOTO 460
       GOTO 85
C
C                    Throw out old derivative information
 27   JSTART = 0
C?      H=HO         ! Use New Step Size (if MAIN changed it)?
      IF ((TO-TOUT)*HO .GE. ZERO) GOTO 420
      GOTO 45
C
 30   IF ((T-TOUT)*H .GE. ZERO) GOTO 450
      IF (MF .NE. MFC) JSTART = -1
      NC = N
      EPSC = EPS
      MFC = MF
      GOTO 45
C
 40   HMAX = HO
      HMAX = AMIN1(HMAX,HMAXMX)
C
C                                  Round-off Warning
```

```
 45   IF ((T+H) .EQ. T) WRITE(LOUT,15) T
C
 50   CALL TSTEP (Y, NO)
C
      KGO = 1 - KFLAG
C     WRITE(LOUT,996) KFLAG
C 996 FORMAT(1H ,2X,'KFLAG='I3)
      GOTO (60, 100, 200, 300), KGO
C KFLAG  =   0,  -1,  -2,  -3  ----------------------------------------
C
 60   CONTINUE
C-----------------------------------------------------------------------
C NORMAL RETURN FROM TSTEP.
C
C THE WEIGHTS YMAX(I) ARE UPDATED.  IF DIFFERENT VALUES ARE DESIRED,
C THEY SHOULD BE SET HERE.  IF SS IS TO BE UPDATED FOR CONTROL OF
C ERROR PER SS UNITS OF T, IT SHOULD ALSO BE DONE HERE.  A TEST IS
C MADE TO DETERMINE IF EPS IS TOO SMALL FOR MACHINE PRECISION.
C
C ANY OTHER TESTS OR CALCULATIONS THAT ARE REQUIRED AFTER EACH STEP
C SHOULD BE INSERTED HERE.
C
C IF INDEX = 3, YO IS SET TO THE CURRENT Y VALUES ON RETURN.
C IF INDEX = 2, H IS CONTROLLED TO HIT TOUT (WITHIN ROUNDOFF
C ERROR), AND THEN THE CURRENT Y VALUES ARE PUT IN YO ON
C RETURN.  FOR ANY OTHER VALUE OF INDEX, CONTROL RETURNS TO
C THE INTEGRATOR UNLESS TOUT HAS BEEN REACHED.  THEN
C INTERPOLATED VALUES OF Y ARE COMPUTED AND STORED IN YO ON
C RETURN.
C IF INTERPOLATION IS NOT DESIRED, THE CALL TO INTERP SHOULD
C BE DELETED AND CONTROL TRANSFERRED TO STATEMENT 500 INSTEAD
C OF 520.
C-----------------------------------------------------------------------
C
C     DO 990 I=1,N
C     IF(Y(I,1).LE.1.0E-20) Y(I,1)=1.0E-20
C 990 CONTINUE
C
      D = ZERO
C                   Initialize to no negative problem
      NFLAG = 0
      DO 70 I = 1,N
        AYI = ABS(Y(I,1))
   GOTO (70, 62, 68, 64, 68, 63, 67, 63, 67), IERROR
C IERROR  =    1,  2,  3,  4,  5,  6,  7,  8,  9 ------- -DRW ---------
C
C                           Relative Error
  62    YMAX(I) = AYI
C                           No sense in permitting /0.
        IF (AYI.EQ.ZERO) YMAX(I)=YMIN
   GOTO 70
  63  IF (Y(I,1).LT.YCUT.AND.I.GE.NRMIN.AND.I.LE.NRMAX) NFLAG=I
C                  Relative Error not below YMIN -DRW
```

```
  64    YMAX(I) = AMAX1(AYI,YMIN)
        GOTO 70
  67  IF (Y(I,1).LT.YCUT.AND.I.GE.NRMIN.AND.I.LE.NRMAX) NFLAG=I
C                            SemiRelative Error
  68    YMAX(I) = AMAX1(YMAX(I), AYI)
        GOTO 70
  70  D = D + (AYI/YMAX(I))**2
      D = D*(UROUND/EPS)**2
C                                         Halt Condition
      IF (D .GT. FLOAT(N)) GOTO 250
      IF (INDEX .EQ. 3) GOTO 500
      IF (INDEX .EQ. 2) GOTO 85
C                                         Integration Passed TOUT
 80   IF ((T-TOUT)*H .GE. ZERO) GOTO 82
C                            Negative Value Error
      IF (NFLAG.GT.0) GOTO 275
C                            Keep Going in Time
      GOTO 45
C                            Passed TOUT, set YO
 82   CALL INTERP (TOUT, Y, NO, YO)
C                            Done, so TO=TOUT
      TO = TOUT
      GOTO 520
 85   IF (((T+H)-TOUT)*H .LE. ZERO) GOTO 45
      IF (ABS(T-TOUT) .LE. HUNDRD*UROUND*HMAX) GOTO 500
      IF ((T-TOUT)*H .GE. ZERO) GOTO 500
      H = (TOUT - T)*(ONE - FOUR*UROUND)
      JSTART = -1
      GOTO 45
C-----------------------------------------------------------------------
C ON AN ERROR RETURN FROM TSTEP, AN IMMEDIATE RETURN OCCURS IF
C KFLAG = -2, AND RECOVERY ATTEMPTS ARE MADE OTHERWISE.
C TO RECOVER, H AND HMIN ARE REDUCED BY A FACTOR OF .1 UP TO 10
C TIMES BEFORE GIVING UP.
C-----------------------------------------------------------------------
 100  WRITE (LOUT,101)
 101  FORMAT (/46H---  MESSAGE FROM SUBROUTINE DRIVE IN EPISODE,,
     1        24H THE O.D.E. SOLVER.  ---/)
      WRITE(LOUT,105) T,HMIN
 105  FORMAT(//35H KFLAG = -1 FROM INTEGRATOR AT T = ,1PE16.6/
     1        40H  ERROR TEST FAILED WITH ABS(H) = HMIN =,1PE16.6/)
 110  IF (NHCUT .EQ. 10) GOTO 150
      NHCUT = NHCUT + 1
      HMIN = HCUT*HMIN
      H = HCUT*H
      WRITE (LOUT,115) H
 115  FORMAT(24H  H HAS BEEN REDUCED TO ,1PE16.6,
     1        26H  AND STEP WILL BE RETRIED//)
      JSTART = -1
      GOTO 45
C
 150  WRITE (LOUT,155)
 155  FORMAT(//44H PROBLEM APPEARS UNSOLVABLE WITH GIVEN INPUT//)
```

```
      GOTO 500
C
 200  WRITE (LOUT,101)
      WRITE (LOUT,205) T,H,EPS
 205  FORMAT(//14H KFLAG= -2  T=,1PE17.7,4H H =,E16.6,6H EPS =,E16.6/
     1         50H  THE REQUESTED ERROR IS TOO SMALL FOR INTEGRATOR.//)
      GOTO 500
C
 250  WRITE (LOUT,101)
      WRITE (LOUT,255) T,EPS
 255  FORMAT(//46H INTEGRATION HALTED BY SUBROUTINE DRIVE AT T =,
     1      1PE17.8/43H EPS IS TOO SMALL FOR MACHINE PRECISION AND/
     2      29H PROBLEM BEING SOLVED.  EPS =,1PE16.6//)
      KFLAG = -2
      GOTO 500
C
 275  WRITE (LOUT,280) T,NFLAG,Y(NFLAG,1)
 280  FORMAT(' INTEGRATION SUSPENDED BY NEGATIVE CONCENTRATION AT',
     $' T=',1PE10.3/' ELEMENT #',I3,' WAS',1PE12.3,6X,'(DRIVES)')
      KFLAG=-7
C                              INDEX for Negative Value
      GOTO 500
C
 300  WRITE (LOUT,101)
      WRITE (LOUT,305) T
 305  FORMAT(//34H KFLAG = -3 FROM INTEGRATOR AT T =,1PE18.8/
     1         45H  CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED/)
      GOTO 110
C
 400  WRITE (LOUT,101)
      WRITE (LOUT,405) EPS
 405  FORMAT(//35H ILLEGAL INPUT.. EPS .LE. 0. EPS = ,E16.6//)
      INDEX = -4
      RETURN
C
 410  WRITE (LOUT,101)
      WRITE (LOUT,415) N
 415  FORMAT(//31H ILLEGAL INPUT.. N .LE. 0. N = ,I8//)
      INDEX = -4
      RETURN
C
 420  WRITE (LOUT,101)
      WRITE (LOUT,425) TO,TOUT,HO
 425  FORMAT(//39H ILLEGAL INPUT.. (TO - TOUT)*HO .GE. 0./
     1         5H TO =,1PE18.8,7H TOUT =,E18.8,5H HO =,E16.6//)
      INDEX = -4
      RETURN
C
 430  WRITE (LOUT,101)
      WRITE (LOUT,435) INDEX
 435  FORMAT(//24H ILLEGAL INPUT.. INDEX =,I8//)
      INDEX = -4
      RETURN
```

```
C
 440  WRITE (LOUT,101)
      WRITE (LOUT,445) N
 445  FORMAT (//39H ILLEGAL INPUT.  THE NUMBER OF ORDINARY/
     1          43H DIFFERENTIAL EQUATIONS BEING SOLVED IS N =, I6/
     2          42H STORAGE ALLOCATION IN SUBROUTINE DRIVE IS/
     3          46H TOO SMALL.  SEE COMMENTS IN SUBROUTINE DRIVE./)
      INDEX = -4
      RETURN
C
 450  WRITE (LOUT,101)
      WRITE (LOUT,455) T,TOUT,H
 455  FORMAT(//46H INDEX = -1 ON INPUT WITH (T - TOUT)*H .GE. 0./
     1        44H INTERPOLATION WAS DONE AS ON NORMAL RETURN./
     2        41H DESIRED PARAMETER CHANGES WERE NOT MADE./
     3        4H T =,E18.8,7H TOUT =,E18.8,4H H =,E16.6//)
      CALL INTERP (TOUT, Y, NO, YO)
      TO = TOUT
      INDEX = -5
      RETURN
C
 460  WRITE (LOUT,101)
      WRITE (LOUT,465) T,TOUT,H
 465  FORMAT(//45H INDEX = 2 ON INPUT WITH (T - TOUT)*H .GE. 0./
     1        4H T =,E18.8,7H TOUT =,E18.8,4H H =,E16.6//)
      INDEX = -6
      RETURN
C
 500  TO = T
      DO 510 I = 1,N
 510    YO(I) = Y(I,1)
 520  INDEX = KFLAG
      TOP = TO
      HO = HUSED
      IF (KFLAG .NE. 0) HO = H
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE INTERP (TOUT, Y, NO, YO)
C---------------------------------------------------------------------
C SUBROUTINE INTERP COMPUTES INTERPOLATED VALUES OF THE DEPENDENT
C VARIABLE Y AND STORES THEM IN YO.  THE INTERPOLATION IS TO THE
C POINT T = TOUT AND USES THE NORDSIECK HISTORY ARRAY Y AS FOLLOWS..
C                                NQ
C                   YO(I)  =  SUM  Y(I,J+1)*S**J ,
C                               J=0
C WHERE S = -(T-TOUT)/H.
C---------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE INTERP.
C---------------------------------------------------------------------
C CAUTION:  NOT ALL MEMBERS OF EPCOM1 ARE USED IN THIS SUBROUTINE.
```

```
C-----------------------------------------------------------------------
      INTEGER NO
      INTEGER JSTART, KFLAG, MF, N
      INTEGER I, J, L
      REAL TOUT, Y, YO
      REAL EPS, H, HMAX, HMIN, SS, T, UROUND
      REAL S, S1
      REAL ONE
      DIMENSION YO(NO),Y(NO,13)
C
      COMMON /EPCOM1/ T,H,HMIN,HMAX,EPS,SS,UROUND,N,MF,KFLAG,JSTART
      DATA ONE /1.0E0/
      DO 10 I = 1,N
 10     YO(I) = Y(I,1)
      L = JSTART + 1
      S = (TOUT - T)/H
      S1 = ONE
      DO 30 J = 2,L
        S1 = S1*S
        DO 20 I = 1,N
 20       YO(I) = YO(I) + S1*Y(I,J)
 30     CONTINUE
      RETURN
      END
C
C=======================================================================
C
      SUBROUTINE TSTEP (Y, NO)
C-----------------------------------------------------------------------
C TSTEP PERFORMS ONE STEP OF THE INTEGRATION OF AN INITIAL VALUE
C PROBLEM FOR A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS.
C COMMUNICATION WITH TSTEP IS VIA THE FOLLOWING VARIABLES..
C
C   Y       AN NO BY LMAX ARRAY CONTAINING THE DEPENDENT VARIABLES
C             AND THEIR SCALED DERIVATIVES.  LMAX IS CURRENTLY 6 FOR
C             THE VARIABLE STEP BACKWARD DIFFERENTIATION FORMULAS,
C             AND 13 FOR THE VARIABLE STEP ADAMS FORMULAS.
C             (LMAX -1) = MAXDER, THE MAXIMUM ORDER USED.
C             SEE SUBROUTINE COSET.  Y(I,J+1) CONTAINS THE
C             J-TH DERIVATIVE OF Y(I), SCALED BY H**J/FACTORIAL(J)
C             FOR J = 0,1,...,NQ, WHERE NQ IS THE CURRENT ORDER.
C   NO      A CONSTANT INTEGER .GE. N, USED FOR DIMENSIONING
C             PURPOSES.
C   T       THE INDEPENDENT VARIABLE, UPDATED ON EACH STEP TAKEN.
C   H       THE STEP SIZE TO BE ATTEMPTED ON THE NEXT STEP.
C             H IS ALTERED BY THE ERROR CONTROL ALGORITHM DURING
C             THE SOLUTION OF THE PROBLEM. H CAN BE EITHER POSITIVE
C             OR NEGATIVE, BUT ITS SIGN MUST REMAIN CONSTANT
C             THROUGHOUT THE PROBLEM RUN.
C   HMIN,   THE MINIMUM AND MAXIMUM ABSOLUTE VALUES OF THE STEP
C    HMAX     SIZE TO BE USED FOR THE STEP. THESE MAY BE CHANGED AT
C             ANY TIME, BUT THE CHANGE WILL NOT TAKE EFFECT UNTIL THE
C             NEXT CHANGE IN H IS MADE.
```

```
C   EPS     THE RELATIVE ERROR BOUND.  SEE DESCRIPTION IN
C             SUBROUTINE DRIVE.
C   SS      THE SIZE OF THE TIME INTERVAL TO BE USED FOR ERROR
C             CONTROL.  A DEFAULT VALUE OF 0 IS USED TO PRODUCE
C             CONTROL OF ERROR PER STEP.  SEE SUBROUTINE DRIVE.
C   UROUND  THE UNIT OF ROUNDOFF FOR THE COMPUTER BEING USED.
C   N       THE NUMBER OF FIRST ORDER ORDINARY DIFFERENTIAL
C             EQUATIONS BEING SOLVED.
C   MF      THE METHOD FLAG.  SEE DESCRIPTION IN SUBROUTINE DRIVE.
C   KFLAG   A COMPLETION CODE WITH THE FOLLOWING MEANINGS..
C                     0  THE STEP WAS SUCCESFUL.
C                    -1  THE REQUESTED ERROR COULD NOT BE ACHIEVED
C                          WITH ABS(H) = HMIN.
C                    -2  THE REQUESTED ERROR IS SMALLER THAN CAN
C                          BE HANDLED FOR THIS PROBLEM.
C                    -3  CORRECTOR CONVERGENCE COULD NOT BE
C                          ACHIEVED FOR ABS(H) = HMIN.
C             ON A RETURN WITH KFLAG NEGATIVE, THE VALUES OF T AND
C             THE Y ARRAY ARE AS OF THE BEGINNING OF THE LAST
C             STEP AND H IS THE LAST STEP SIZE ATTEMPTED.
C   JSTART  AN INTEGER USED ON INPUT AND OUTPUT.
C             ON INPUT, IT HAS THE FOLLOWING VALUES AND MEANINGS..
C                     0  PERFORM THE FIRST STEP.
C                 .GT.0  TAKE A NEW STEP CONTINUING FROM THE LAST.
C                 .LT.0  TAKE THE NEXT STEP WITH A NEW VALUE OF
C                          H AND/OR MF.
C             ON EXIT, JSTART IS SET TO NQ, THE CURRENT ORDER OF THE
C             METHOD.
C   YMAX      AN ARRAY OF N ELEMENTS WITH WHICH THE ESTIMATED LOCAL
C               ERRORS IN Y ARE COMPARED.
C   ERROR     AN ARRAY OF N ELEMENTS.  ERROR(I)/TQ(2) IS THE
C               ESTIMATED LOCAL ERROR IN Y(I) PER SS UNITS OF
C               T OR PER STEP (OF SIZE H).
C   SAVE1,    TWO ARRAYS FOR WORKING STORAGE,
C     SAVE2     EACH OF LENGTH N.
C   PW        A BLOCK OF LOCATIONS USED FOR THE PARTIAL DERIVATIVES
C               OF F WITH RESPECT TO Y, IF MITER IS NOT 0.  SEE
C               DESCRIPTION IN SUBROUTINE DRIVE.
C   IPIV      AN INTEGER ARRAY OF LENGTH N, WHICH IS USED FOR PIVOT
C               INFORMATION FOR THE LINEAR ALGEBRAIC SYSTEM IN THE
C               CORRECTION PROCESS, WHEN MITER = 1 OR 2.
C
C THE COMMON BLOCK EPCM10, DECLARED BELOW, IS PRIMARILY INTENDED
C FOR INTERNAL USE, BUT IT CAN BE ACCESSED EXTERNALLY.
C-----------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE TSTEP.
C-----------------------------------------------------------------------
      PARAMETER ( NMAX =120 )
      PARAMETER ( NMAXSQ = NMAX*NMAX )
      INTEGER NO
      INTEGER IPIV, JSTART, KFLAG, L, LMAX, METH, MF, N, NFE, NJE,
     1        NQ, NQINDX, NQUSED, NSTEP
      INTEGER I, IBACK, IER, IREDO, J, J1, J2, M, MFOLD, MIO,
     1         MITER, MITER1, NEWJ, NSTEPJ
      INTEGER ISTEPJ, KFC, KFH, MAXCOR
      REAL Y
      REAL EL, EPS, ERROR, H, HMAX, HMIN, HUSED, PW,
     1     SAVE1, SAVE2, SS, T, TAU, TQ, UROUND, YMAX
      REAL BND, CNQUOT, CON, CONP, CRATE, D, DRC,
     1     D1, E, EDN, ETA, ETAMAX, ETAMIN, ETAQ, ETAQM1,
     2     ETAQP1, EUP, FLOTL, FLOTN, HOLD, HRL1, PHRL1,
     3     PRL1, R, RC, RL1, R0, R1, TOLD
      REAL ADDON, BIAS1, BIAS2, BIAS3, CRDOWN, DELRC,
     1     ETACF, ETAMXF, ETAMX1, ETAMX2,
     2     ETAMX3, ONEPSM, SHORT, THRESH
      REAL ONE, PT5, ZERO
C* Multiple Declaration of ETAMIN fixed - DRW
      DIMENSION Y(N0,13)
C
      COMMON /EPCOM1/ T,H,HMIN,HMAX,EPS,SS,UROUND,N,MF,KFLAG,JSTART
      COMMON /EPCOM2/ YMAX(NMAX)
      COMMON /EPCOM3/ ERROR(NMAX)
      COMMON /EPCOM4/ SAVE1(NMAX)
      COMMON /EPCOM5/ SAVE2(NMAX)
      COMMON /EPCOM6/ PW(NMAXSQ)
      COMMON /EPCOM7/ IPIV(NMAX)
      COMMON /EPCOM9/ HUSED,NQUSED,NSTEP,NFE,NJE
      COMMON /EPCM10/ TAU(13),EL(13),TQ(5),LMAX,METH,NQ,L,NQINDX
      COMMON /EPCO99/ NCSTEP,NCFE,NCJE
C
      DATA ISTEPJ /20/, KFC /-3/, KFH /-7/, MAXCOR /3/
      DATA ADDON  /1.0E-6/,     BIAS1  /2.5E1/, BIAS2  /2.5E1/,
     1     BIAS3  /1.0E2/,      CRDOWN /0.1E0/, DELRC  /0.3E0/,
     2     ETACF  /0.25E0/,     ETAMIN /0.1E0/, ETAMXF /0.2E0/,
     3     ETAMX1 /1.0E4/,      ETAMX2 /1.0E1/, ETAMX3 /1.5E0/,
     4     ONEPSM /1.00001E0/, SHORT  /0.1E0/, THRESH /1.3E0/
      DATA ONE /1.0E0/, PT5 /0.5E0/, ZERO /0.0E0/
      KFLAG = 0
      TOLD = T
      FLOTN = FLOAT(N)
      IF (JSTART .GT. 0) GOTO 200
      IF (JSTART .NE. 0) GOTO 150
C-----------------------------------------------------------------------
C ON THE FIRST CALL, THE ORDER IS SET TO 1 AND THE INITIAL
C DERIVATIVES ARE CALCULATED.  ETAMAX IS THE MAXIMUM RATIO BY
C WHICH H CAN BE INCREASED IN A SINGLE STEP.  IT IS 1.E04 FOR THE
C FIRST STEP TO COMPENSATE FOR THE SMALL INITIAL H, THEN 10 FOR
C THE NEXT 10 STEPS, AND THEN 1.5 THEREAFTER.  IF A FAILURE
C OCCURS (IN CORRECTOR CONVERGENCE OR ERROR TEST), ETAMAX IS SET AT 1
C FOR THE NEXT INCREASE.  ETAMIN = .1 IS THE MINIMUM RATIO BY WHICH
C H CAN BE REDUCED ON ANY RETRY OF A STEP.
C-----------------------------------------------------------------------
      CALL DIFFUN (N, T, Y, SAVE1)
      DO 110 I = 1,N
 110    Y(I,2) = H*SAVE1(I)
      METH = MF/10
      MITER = MF - 10*METH
      MITER1 = MITER + 1
      MFOLD = MF
      NQ = 1
      L = 2
      TAU(1) = H
      PRL1 = ONE
      RC = ZERO
      ETAMAX = ETAMX1
      NQINDX = 2
C          For unknown reasons, these variables are sometimes
C          unitialized and cause the program to crash under
C          Microsoft FORTRAN, despite a BLOCK DATA initialization.
C     WRITE(99,789) NSTEP,NCSTEP
C789  FORMAT(' NSTEP=',I8,'   NCSTEP=',I8)
C                               Cumulative Values
      NCSTEP=NCSTEP+NSTEP
      NCFE=NCFE+NFE
      NCJE=NCJE+NJE
      NSTEP = 0
      NSTEPJ = 0
      NFE = 1
      NJE = 0
      GOTO 200
C-----------------------------------------------------------------------
C IF THE USER HAS CHANGED H, THEN Y MUST BE RESCALED.  IF THE
C USER HAS CHANGED MITER, THEN NEWJ IS SET TO MITER TO FORCE
C THE PARTIAL DERIVATIVEES TO BE UPDATED, IF THEY ARE BEING USED.
C-----------------------------------------------------------------------
 150  IF (MF .EQ. MFOLD) GOTO 170
      MIO = MITER
      METH = MF/10
      MITER = MF - 10*METH
      MFOLD = MF
      IF (MITER .EQ. MIO) GOTO 170
      NEWJ = MITER
      MITER1 = MITER + 1
 170  IF (H .EQ. HOLD) GOTO 200
      ETA = H/HOLD
      H = HOLD
      IREDO = 3
      GOTO 185
  180 ETA = AMAX1(ETA,HMIN/ABS(H),ETAMIN)
  185 ETA = AMIN1(ETA,HMAX/ABS(H),ETAMAX)
      R1 = ONE
      DO 190 J = 2,L
        R1 = R1*ETA
        DO 190 I = 1,N
 190      Y(I,J) = Y(I,J)*R1
      H = H*ETA
      RC = RC*ETA
      IF (IREDO .EQ. 0) GOTO 690
C-----------------------------------------------------------------------
```

```
C THIS SECTION COMPUTES THE PREDICTED VALUES BY EFFECTIVELY
C MULTIPLYING THE Y ARRAY BY THE PASCAL TRIANGLE MATRIX.  THEN
C COSET IS CALLED TO OBTAIN EL, THE VECTOR OF COEFFICIENTS OF
C LENGTH NQ + 1.  RC IS THE RATIO OF NEW TO OLD VALUES OF THE
C COEFFICIENT H/EL(2).  WHEN RC DIFFERS FROM 1 BY MORE THAN
C DELRC, NEWJ IS SET TO MITER TO FORCE THE PARTIAL DERIVATIVES
C TO BE UPDATED, IF USED.  DELRC IS 0.3.  IN ANY CASE, THE PARTIAL
C DERIVATIVES ARE UPDATED AT LEAST EVERY 20-TH STEP.
C-----------------------------------------------------------------------
 200  T = T + H
      DO 210 J1 = 1,NQ
        DO 210 J2 = J1,NQ
          J = (NQ + J1) - J2
          DO 210 I = 1,N
 210        Y(I,J) = Y(I,J) + Y(I,J+1)
      CALL COSET
      BND = FLOTN*(TQ(4)*EPS)**2
      RL1 = ONE/EL(2)
      RC = RC*(RL1/PRL1)
      PRL1 = RL1
      IF (NSTEP .GE. NSTEPJ+ISTEPJ) NEWJ = MITER
      DRC = ABS(RC-ONE)
      IF (DRC .LE. DELRC) GOTO 215
      NEWJ = MITER
      CRATE = ONE
      RC = ONE
      GOTO 220
  215 IF ((MITER .NE. 0) .AND. (DRC .NE. ZERO)) CRATE = ONE
C-----------------------------------------------------------------------
C UP TO 3 CORRECTOR ITERATIONS ARE TAKEN.  A CONVERGENCE TEST IS MADE
C ON THE ROOT MEAN SQUARE NORM OF EACH CORRECTION, USING BND, WHICH
C IS DEPENDENT ON EPS.  THE SUM OF THE CORRECTIONS IS ACCUMULATED IN
C THE VECTOR ERROR.  THE Y ARRAY IS NOT ALTERED IN THE CORRECTOR
C LOOP.  THE UPDATED Y VECTOR IS STORED TEMPORARILY IN SAVE1.
C-----------------------------------------------------------------------
 220  DO 230 I = 1,N
 230    ERROR(I) = ZERO
      M = 0
      CALL DIFFUN (N, T, Y, SAVE2)
      NFE = NFE + 1
      IF (NEWJ .LE. 0) GOTO 290
C-----------------------------------------------------------------------
C IF INDICATED, THE MATRIX P = I - H*RL1*J IS REEVALUATED BEFORE
C STARTING THE CORRECTOR ITERATION.  NEWJ IS SET TO 0 AS AN
C INDICATOR THAT THIS HAS BEEN DONE.  IF MITER = 1 OR 2, P IS
C COMPUTED AND PROCESSED IN PSET.  IF MITER = 3, THE MATRIX  IS
C P = I - H*RL1*D, WHERE D IS A DIAGONAL MATRIX.  RL1 IS 1/EL(2).
C-----------------------------------------------------------------------
      NEWJ = 0
      RC = ONE
      NJE = NJE + 1
      NSTEPJ = NSTEP
      GOTO (250, 240, 260), MITER
 240  NFE = NFE + N
 250  CON = -H*RL1
      CALL PSET(Y, NO, CON, MITER, IER)
      IF (IER .NE. 0) GOTO 420
      GOTO 350
 260  R = RL1*SHORT
      DO 270 I = 1,N
 270    PW(I) = Y(I,1) + R*(H*SAVE2(I) - Y(I,2))
      CALL DIFFUN(N, T, PW, SAVE1)
      NFE = NFE + 1
      HRL1 = H*RL1
      DO 280 I = 1,N
        RO = H*SAVE2(I) - Y(I,2)
        PW(I) = ONE
        D = SHORT*RO - H*(SAVE1(I) - SAVE2(I))
        SAVE1(I) = ZERO
        IF (ABS(RO) .LT. UROUND*YMAX(I)) GOTO 280
        IF (ABS(D) .EQ. ZERO) GOTO 420
        PW(I) = SHORT*RO/D
        SAVE1(I) = PW(I)*RL1*RO
 280    CONTINUE
      GOTO 370
 290  GOTO (295, 350, 350, 310), MITER1
C-----------------------------------------------------------------------
C IN THE CASE OF FUNCTIONAL ITERATION, Y IS UPDATED DIRECTLY FROM
C THE RESULT OF THE LAST DIFFUN CALL.
C-----------------------------------------------------------------------
 295  D = ZERO
      DO 300 I = 1,N
C
C         WRITE(LOUT,999) I,YMAX(I)
C 999     FORMAT(1H 2X,'I=',I2,2X,'YMAX=',E12.5)
C
         R = RL1*(H*SAVE2(I) - Y(I,2))
C
C  The next line often gave an Undefined Real Error.
C  Let's try to fix it without changing anything else.  - DRW
C
C         D = D + ((R - ERROR(I))/YMAX(I))**2
C
         DTERM = ABS ( (R-ERROR(I)) / YMAX(I) )
         IF (DTERM.LT.1.E15) DTERM = DTERM*DTERM
         D = D + DTERM
         SAVE1(I) = Y(I,1) + R
 300     ERROR(I) = R
      GOTO 400
C-----------------------------------------------------------------------
C IN THE CASE OF A CHORD METHOD, THE RESIDUAL -G(Y SUB N(M))
C IS COMPUTED AND THE LINEAR SYSTEM WITH THAT AS RIGHT-HAND SIDE
C AND P AS COEFFICIENT MATRIX IS SOLVED, USING THE LU DECOMPOSITION
C OF P IF MITER = 1 OR 2.  IF MITER = 3 THE SCALAR H*RL1 IS UPDATED.
C-----------------------------------------------------------------------
 310  PHRL1 = HRL1
```

```
      HRL1 = H*RL1
      IF (HRL1 .EQ. PHRL1) GOTO 330
      R = HRL1/PHRL1
      DO 320 I = 1,N
        D = ONE - R*(ONE - ONE/PW(I))
        IF (ABS(D) .EQ. ZERO) GOTO 440
 320    PW(I) = ONE/D
 330  DO 340 I = 1,N
 340    SAVE1(I) = PW(I)*(RL1*H*SAVE2(I) - (RL1*Y(I,2) + ERROR(I)))
      GOTO 370
 350  DO 360 I = 1,N
 360    SAVE1(I) = RL1*H*SAVE2(I) - (RL1*Y(I,2) + ERROR(I))
      CALL SOL (N, NO, PW, SAVE1, IPIV)
 370  D = ZERO
      DO 380 I = 1,N
        ERROR(I) = ERROR(I) + SAVE1(I)
 380    SAVE1(I) = Y(I,1) + ERROR(I)
C-----------------------------------------------------------------------
C TEST FOR CONVERGENCE.  IF M .GT. 0, AN ESTIMATE OF THE SQUARE OF
C THE CONVERGENCE RATE CONSTANT IS STORED IN CRATE, AND THIS IS USED
C IN THE TEST.
C-----------------------------------------------------------------------
  400 IF (M .NE. 0) CRATE = AMAX1(CRDOWN*CRATE,D/D1)
      IF (D*AMIN1(ONE,CRATE) .LE. BND) GOTO 450
      D1 = D
      M = M + 1
      IF (M .EQ. MAXCOR) GOTO 410
      CALL DIFFUN (N, T, SAVE1, SAVE2)
      GOTO (295, 350, 350, 310), MITER1
C-----------------------------------------------------------------------
C THE CORRECTOR ITERATION FAILED TO CONVERGE IN 3 TRIES. IF PARTIAL
C DERIVATIVES ARE INVOLVED BUT ARE NOT UP TO DATE, THEY ARE
C REEVALUATED FOR THE NEXT TRY.  OTHERWISE THE Y ARRAY IS RESTORED
C TO ITS VALUES BEFORE PREDICTION, AND H IS REDUCED,
C IF POSSIBLE.  IF NOT, A NO-CONVERGENCE EXIT IS TAKEN.
C-----------------------------------------------------------------------
 410  NFE = NFE + MAXCOR - 1
      IF (NEWJ .EQ. -1) GOTO 440
 420  T = TOLD
      ETAMAX = ONE
      DO 430 J1 = 1,NQ
        DO 430 J2 = J1,NQ
          J = (NQ + J1) - J2
          DO 430 I = 1,N
 430        Y(I,J) = Y(I,J) - Y(I,J+1)
      IF (ABS(H) .LE. HMIN*ONEPSM) GOTO 680
      ETA = ETACF
      IREDO = 1
      GOTO 180
 440  NEWJ = MITER
      GOTO 220
C-----------------------------------------------------------------------
C THE CORRECTOR HAS CONVERGED.  NEWJ IS SET TO -1 IF PARTIAL
```

```
C DERIVATIVES WERE USED, TO SIGNAL THAT THEY MAY NEED UPDATING ON
C SUBSEQUENT STEPS.  THE ERROR TEST IS MADE AND CONTROL PASSES TO
C STATEMENT 500 IF IT FAILS.
C-----------------------------------------------------------------------
 450  IF (MITER .NE. 0) NEWJ = -1
      NFE = NFE + M
      D = ZERO
      DO 460 I = 1,N
 460    D = D + (ERROR(I)/YMAX(I))**2
      E = FLOTN*(TQ(2)*EPS)**2
      IF (D .GT. E) GOTO 500
C-----------------------------------------------------------------------
C AFTER A SUCCESSFUL STEP, THE Y ARRAY, TAU, NSTEP, AND NQINDX ARE
C UPDATED, AND A NEW VALUE OF H AT ORDER NQ IS COMPUTED.
C THE VECTOR TAU CONTAINS THE NQ + 1 MOST RECENT VALUES OF H.
C A CHANGE IN NQ UP OR DOWN BY 1 IS CONSIDERED IF NQINDX = 0.
C IF NQINDX = 1 AND NQ .LT. MAXDER, THEN ERROR IS SAVED
C FOR USE IN A POSSIBLE ORDER INCREASE ON THE NEXT STEP.
C A CHANGE IN H OR NQ IS MADE ONLY OF THE INCREASE IN H
C IS BY A FACTOR OF AT LEAST 1.3.
C IF NOT, NQINDX IS SET TO 2 TO PREVENT TESTING FOR THAT MANY
C STEPS.  IF NQ IS CHANGED, NQINDX IS SET TO NQ + 1 (NEW VALUE).
C-----------------------------------------------------------------------
      KFLAG = 0
      IREDO = 0
      NSTEP = NSTEP + 1
      HUSED = H
      NQUSED = NQ
      DO 470 IBACK = 1,NQ
        I = L - IBACK
 470    TAU(I+1) = TAU(I)
      TAU(1) = H
      DO 480 J = 1,L
        DO 480 I = 1,N
 480      Y(I,J) = Y(I,J) + ERROR(I)*EL(J)
      NQINDX = NQINDX - 1
      IF ((L .EQ. LMAX) .OR. (NQINDX .NE. 1)) GOTO 495
      DO 490 I = 1,N
 490    Y(I,LMAX) = ERROR(I)
      CONP = TQ(5)
 495  IF (ETAMAX .NE. ONE) GOTO 520
      IF (NQINDX .LT. 2) NQINDX = 2
      GOTO 690
C-----------------------------------------------------------------------
C THE ERROR TEST FAILED.  KFLAG KEEPS TRACK OF MULTIPLE FAILURES.
C T AND THE Y ARRAY ARE RESTORED TO THEIR PREVIOUS VALUES.  A NEW
C H FOR A RETRY OF THE STEP IS COMPUTED.  THE ORDER IS KEPT FIXED.
C-----------------------------------------------------------------------
 500  KFLAG = KFLAG - 1
      T = TOLD
      DO 510 J1 = 1,NQ
        DO 510 J2 = J1,NQ
          J = (NQ + J1) - J2
```

```
         DO 510 I = 1,N
 510       Y(I,J) = Y(I,J) - Y(I,J+1)
      NEWJ = MITER
      ETAMAX = ONE
      IF (ABS(H) .LE. HMIN*ONEPSM) GOTO 660
      IF (KFLAG .LE. KFC) GOTO 630
      IREDO = 2
C COMPUTE RATIO OF NEW H TO CURRENT H AT THE CURRENT ORDER. ------------
  520 FLOTL = FLOAT(L)
      ETAQ = ONE/((BIAS2*D/E)**(PT5/FLOTL) + ADDON)
      IF ((NQINDX .NE. 0) .OR. (KFLAG .NE. 0)) GOTO 580
      ETAQM1 = ZERO
      IF (NQ .EQ. 1) GOTO 540
C COMPUTE RATIO OF NEW H TO CURRENT H AT THE CURRENT ORDER LESS ONE. ---
      D = ZERO
      DO 530 I = 1,N
 530    D = D + (Y(I,L)/YMAX(I))**2
      EDN = FLOTN*(TQ(1)*EPS)**2
      ETAQM1 = ONE/((BIAS1*D/EDN)**(PT5/(FLOTL - ONE)) + ADDON)
 540  ETAQP1 = ZERO
      IF (L .EQ. LMAX) GOTO 560
C COMPUTE RATIO OF NEW H TO CURRENT H AT CURRENT ORDER PLUS ONE. -------
      CNQUOT = (TQ(5)/CONP)*(H/TAU(2))**L
      D = ZERO
      DO 550 I = 1,N
 550    D = D + ((ERROR(I) - CNQUOT*Y(I,LMAX))/YMAX(I))**2
      EUP = FLOTN*(TQ(3)*EPS)**2
      ETAQP1 = ONE/((BIAS3*D/EUP)**(PT5/(FLOTL + ONE)) + ADDON)
 560  NQINDX = 2
      IF (ETAQ .GE. ETAQP1) GOTO 570
      IF (ETAQP1 .GT. ETAQM1) GOTO 600
      GOTO 590
 570  IF (ETAQ .LT. ETAQM1) GOTO 590
 580  IF ((ETAQ .LT. THRESH) .AND. (KFLAG .EQ. 0)) GOTO 690
      ETA = ETAQ
      IF ((KFLAG .LE. -2) .AND. (ETA .GT. ETAMXF)) ETA = ETAMXF
      GOTO 180
 590  IF (ETAQM1 .LT. THRESH) GOTO 690
      CALL ADJUST (Y, NO)
      L = NQ
      NQ = NQ - 1
      ETA = ETAQM1
      NQINDX = L
      GOTO 180
 600  IF (ETAQP1 .LT. THRESH) GOTO 690
      NQ = L
      ETA = ETAQP1
      L = L + 1
      DO 610 I = 1,N
 610    Y(I,L) = ZERO
      NQINDX = L
      GOTO 180
C-----------------------------------------------------------------------
```

```
C CONTROL REACHES THIS SECTION IF 3 OR MORE CONSECUTIVE FAILURES
C HAVE OCCURRED.  IT IS ASSUMED THAT THE ELEMENTS OF THE Y ARRAY
C HAVE ACCUMULATED ERRORS OF THE WRONG ORDER.  THE ORDER IS REDUCED
C BY ONE, IF POSSIBLE.  THEN H IS REDUCED BY A FACTOR OF 0.1 AND
C THE STEP IS RETRIED.  AFTER A TOTAL OF 7 CONSECUTIVE FAILURES,
C AN EXIT IS TAKEN WITH KFLAG = -2.
C-----------------------------------------------------------------------
 630  IF (KFLAG .EQ. KFH) GOTO 670
      IF (NQ .EQ. 1) GOTO 640
      ETA = ETAMIN
      CALL ADJUST (Y, NO)
      L = NQ
      NQ = NQ - 1
      NQINDX = L
      GOTO 180
  640 ETA = AMAX1(ETAMIN,HMIN/ABS(H))
      H = H*ETA
      CALL DIFFUN (N, T, Y, SAVE1)
      NFE = NFE + 1
      DO 650 I = 1,N
 650    Y(I,2) = H*SAVE1(I)
      NQINDX = 10
      GOTO 200
C-----------------------------------------------------------------------
C ALL RETURNS ARE MADE THROUGH THIS SECTION.  H IS SAVED IN HOLD
C TO ALLOW THE CALLER TO CHANGE H ON THE NEXT STEP.
C-----------------------------------------------------------------------
 660  KFLAG = -1
      GOTO 700
 670  KFLAG = -2
      GOTO 700
 680  KFLAG = -3
      GOTO 700
 690  ETAMAX = ETAMX3
      IF (NSTEP .LE. 10) ETAMAX = ETAMX2
 700  HOLD = H
      JSTART = NQ
      RETURN
      END
C
C=======================================================================
C
      SUBROUTINE COSET
C-----------------------------------------------------------------------
C COSET IS CALLED BY TSTEP AND SETS COEFFICIENTS FOR USE THERE.
C
C FOR EACH ORDER NQ, THE COEFFICIENTS IN EL ARE CALCULATED BY USE OF
C  THE GENERATING POLYNOMIAL LAMBDA(X), WITH COEFFICIENTS EL(I):
C      LAMBDA(X) = EL(1) + EL(2)*X + ... + EL(NQ+1)*(X**NQ).
C FOR THE BACKWARD DIFFERENTIATION FORMULAS,
C                        NQ
C      LAMBDA(X) = PRODUCT (1 + X/XI(I) ) .
C                      I = 1
```

```
C FOR THE ADAMS FORMULAS,
C                                 NQ-1
C       (D/DX) LAMBDA(X) = C * PRODUCT (1 + X/XI(I) ) ,
C                                 I = 1
C       LAMBDA(-1) = 0,    LAMBDA(0) = 1,
C WHERE C IS A NORMALIZATION CONSTANT.
C IN BOTH CASES, XI(I) IS DEFINED BY
C       H*XI(I) = T SUB N  -  T SUB (N-I)
C               = H + TAU(1) + TAU(2) + ... TAU(I-1).
C
C COSET ALSO SETS MAXDER, THE MAXIMUM ORDER OF THE FORMULAS
C AVAILABLE. CURRENTLY THIS IS 5 FOR THE BACKWARD DIFFERENTIATION
C FORMULAS, AND 12 FOR THE ADAMS FORMULAS.  TO USE DIFFERENT
C VALUES (.LE. 13),  CHANGE THE NUMBERS IN STATEMENTS 1 AND 2 BELOW.
C
C IN ADDITION TO VARIABLES DESCRIBED PREVIOUSLY, COMMUNICATION
C WITH COSET USES THE FOLLOWING..
C   TAU    = A VECTOR OF LENGTH 13 CONTAINING THE PAST NQ VALUES
C            OF H.
C   EL     = A VECTOR OF LENGTH 13 IN WHICH COSET STORES THE
C            COEFFICIENTS FOR THE CORRECTOR FORMULA.
C   TQ     = A VECTOR OF LENGTH 5 IN WHICH COSET STORES CONSTANTS
C            USED FOR THE CONVERGENCE TEST, THE ERROR TEST, AND
C            SELECTION OF H AT A NEW ORDER.
C   LMAX   = MAXDER + 1, WHERE MAXDER IS THE MAXIMUM ORDER
C            AVAILABLE.  LMAX IS THE MAXIMUM NUMBER OF COLUMNS
C            OF THE Y ARRAY TO BE USED.
C   METH   = THE BASIC METHOD INDICATOR.
C   NQ     = THE CURRENT ORDER.
C   L      = NQ + 1, THE LENGTH OF THE VECTOR STORED IN EL, AND
C            THE NUMBER OF COLUMNS OF THE Y ARRAY BEING USED.
C   NQINDX = A COUNTER CONTROLLING THE FREQUENCY OF ORDER CHANGES.
C            AN ORDER CHANGE IS ABOUT TO BE CONSIDERED IF
C            NQINDX = 1.
C-----------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE COSET.
C-----------------------------------------------------------------------
C CAUTION:  NOT ALL MEMBERS OF EPCOM1 ARE USED IN THIS SUBROUTINE.
C-----------------------------------------------------------------------
      INTEGER JSTART, KFLAG, L, LMAX, METH, MF, N, NQ, NQINDX
      INTEGER I, IBACK, J, JP1, MAXDER, LMAXN, NQM1
      REAL EL, EPS, H, HAX, HMIN, SS, T, TAU, TQ,
     1     UROUND
      REAL AHDSS, CNQM1, CSUM, ELP, EM, EMO, FLOTI,
     1     FLOTL, FLOTNQ, HSUM, HSUM1, PROD, RXI, S, XI
      REAL CORTES
      REAL ONE, SIX, TWO, ZERO
C* Multiple Declaration of JSTART,KFLAG,L,METH,MF,NQ,NQINDX, fixed - DRW
      DIMENSION EM(13)
C
      COMMON /EPCOM1/ T,H,HMIN,HMAX,EPS,SS,UROUND,N,MF,KFLAG,JSTART
      COMMON /EPCM10/ TAU(13),EL(13),TQ(5),LMAX,METH,NQ,L,NQINDX
      DATA CORTES /0.1E0/
      DATA ONE  /1.0E0/, SIX /6.0E0/, TWO /2.0E0/, ZERO /0.0E0/
      AHDSS = ONE
      IF (SS .NE. ZERO) AHDSS = ABS(H)/SS
      FLOTL = FLOAT(L)
      NQM1 = NQ - 1
      GOTO (1, 2), METH
 1    MAXDER = 12
      GOTO 100
C
 2    MAXDER = 5
      GOTO 200
C
 100  IF (NQ .NE. 1) GOTO 110
      EL(1) = ONE
      EL(2) = ONE
      TQ(1) = ONE
      TQ(2) = TWO*AHDSS
      TQ(3) = SIX*TQ(2)
      TQ(5) = ONE
      GOTO 300
 110  HSUM = H
      EM(1) = ONE
      FLOTNQ = FLOTL - ONE
      DO 115 I = 2,L
 115    EM(I) = ZERO
      DO 150 J = 1,NQM1
        IF ((J .NE. NQM1) .OR. (NQINDX .NE. 1)) GOTO 130
        S = ONE
        CSUM = ZERO
        DO 120 I = 1,NQM1
          CSUM = CSUM + S*EM(I)/FLOAT(I+1)
 120      S = -S
        TQ(1) = AHDSS*EM(NQM1)/(FLOTNQ*CSUM)
 130    RXI = H/HSUM
        DO 140 IBACK = 1,J
          I = (J + 2) - IBACK
 140      EM(I) = EM(I) + EM(I-1)*RXI
 150    HSUM = HSUM + TAU(J)
C COMPUTE INTEGRAL FROM -1 TO 0 OF POLYNOMIAL AND OF X TIMES IT. -------
      S = ONE
      EMO = ZERO
      CSUM = ZERO
      DO 160 I = 1,NQ
        FLOTI = FLOAT(I)
        EMO = EMO + S*EM(I)/FLOTI
        CSUM = CSUM + S*EM(I)/(FLOTI+1)
 160    S = -S
C IN EL, FORM COEFFICIENTS OF NORMALIZED INTEGRATED POLYNOMIAL. --------
      S = ONE/EMO
      EL(1) = ONE
      DO 170 I = 1,NQ
 170    EL(I+1) = S*EM(I)/FLOAT(I)
      XI = HSUM/H
      TQ(2) = AHDSS*XI*EMO/CSUM
      TQ(5) = XI/EL(L)
      IF (NQINDX .NE. 1) GOTO 300
C FOR HIGHER ORDER CONTROL CONSTANT, MULTIPLY POLYNOMIAL BY 1+X/XI(Q). -
      RXI = ONE/XI
      DO 180 IBACK = 1,NQ
        I = (L + 1) - IBACK
 180    EM(I) = EM(I) + EM(I-1)*RXI
C COMPUTE INTEGRAL OF POLYNOMIAL. --------------------------------------
      S = ONE
      CSUM = ZERO
      DO 190 I = 1,L
      CSUM = CSUM + S*EM(I)/FLOAT(I+1)
 190    S = -S
      TQ(3) = AHDSS*FLOTL*EMO/CSUM
      GOTO 300
C
 200  DO 210 I = 3,L
 210    EL(I) = ZERO
      EL(1) = ONE
      EL(2) = ONE
      HSUM = H
      HSUM1 = ZERO
      PROD = ONE
      RXI = ONE
      IF (NQ .EQ. 1) GOTO 240
      DO 230 J = 1,NQM1
C IN EL, CONSTRUCT COEFFICIENTS OF (1+X/XI(1))*...*(1+X/XI(J+1)). ------
        HSUM = HSUM + TAU(J)
        HSUM1 = HSUM1 + TAU(J)
        PROD = PROD*(HSUM/HSUM1)
        RXI = H/HSUM
        JP1 = J + 1
        DO 220 IBACK = 1,JP1
          I = (J + 3) - IBACK
 220      EL(I) = EL(I) + EL(I-1)*RXI
 230    CONTINUE
 240  TQ(2) = AHDSS*EL(2)*(ONE + PROD)
      TQ(5) = (ONE + PROD)/EL(L)
      IF (NQINDX .NE. 1) GOTO 300
      CNQM1 = RXI/EL(L)
      ELP = EL(2) - RXI
      TQ(1) = AHDSS*ELP/CNQM1
      HSUM = HSUM + TAU(NQ)
      RXI = H/HSUM
      ELP = EL(2) + RXI
      TQ(3) = AHDSS*ELP*RXI*(ONE + PROD)*(FLOTL + ONE)
 300  TQ(4) = CORTES*TQ(2)
      LMAX = MAXDER + 1
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE ADJUST (Y, NO)
C-----------------------------------------------------------------------
C THIS SUBROUTINE ADJUSTS THE Y ARRAY ON REDUCTION OF ORDER.
C SEE REFERENCE 1 FOR DETAILS.
C-----------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE ADJUST.
C-----------------------------------------------------------------------
C CAUTION:  NOT ALL MEMBERS OF EPCOM1 ARE USED IN THIS SUBROUTINE.
C-----------------------------------------------------------------------
      INTEGER NO
      INTEGER JSTART, KFLAG, L, LMAX, METH, MF, N, NQ, NQINDX
      INTEGER I, IBACK, J, JP1, NQM1, NQM2
      REAL Y
      REAL EL, EPS, H, HMAX, HMIN, SS, T, TAU, TQ, UROUND
      REAL HSUM, XI
      REAL ONE, ZERO
      DIMENSION Y(NO,13)
C
      COMMON /EPCOM1/ T,H,HMIN,HMAX,EPS,SS,UROUND,N,MF,KFLAG,JSTART
      COMMON /EPCM10/ TAU(13),EL(13),TQ(5),LMAX,METH,NQ,L,NQINDX
      DATA ONE /1.0E0/, ZERO /0.0E0/
      IF (NQ .EQ. 2) RETURN
      NQM1 = NQ - 1
      NQM2 = NQ - 2
      GOTO (100, 200), METH
C
 100  DO 110 J = 1,LMAX
 110    EL(J) = ZERO
      EL(2) = ONE
      HSUM = ZERO
      DO 130 J = 1,NQM2
C CONSTRUCT COEFFICIENTS OF X*(X+XI(1))*...*(X+XI(J)). -----------------
        HSUM = HSUM + TAU(J)
        XI = HSUM/H
        JP1 = J + 1
        DO 120 IBACK = 1,JP1
          I = (J + 3) - IBACK
 120      EL(I) = EL(I)*XI + EL(I-1)
 130    CONTINUE
C CONSTRUCT COEFFICIENTS OF INTEGRATED POLYNOMIAL. ---------------------
      DO 140 J = 2,NQM1
 140    EL(J+1) = FLOAT(NQ)*EL(J)/FLOAT(J)
      GOTO 300
C
 200  DO 210 J = 1,LMAX
 210    EL(J) = ZERO
      EL(3) = ONE
      HSUM = ZERO
      DO 230 J = 1,NQM2
C CONSTRUCT COEFFICIENTS OF X*X*(X+XI(1))*...*(X+XI(J)). ---------------
        HSUM = HSUM + TAU(J)
        XI = HSUM/H
        JP1 = J + 1
        DO 220 IBACK = 1,JP1
          I = (J + 4) - IBACK
 220      EL(I) = EL(I)*XI + EL(I-1)
 230    CONTINUE
C
C SUBTRACT CORRECTION TERMS FROM Y ARRAY. ------------------------------
 300  DO 320 J = 3,NQ
        DO 310 I = 1,N
 310      Y(I,J) = Y(I,J) - Y(I,L)*EL(J)
 320    CONTINUE
      RETURN
      END
C
C======================================================================
C
      SUBROUTINE PSET (Y,NO,CON,MITER,IER)
C----------------------------------------------------------------------
C PSET IS CALLED BY TSTEP TO COMPUTE AND TO PROCESS THE MATRIX
C P = I - (H/EL(2))*J, WHERE J IS AN APPROXIMATION TO THE
C JACOBIAN.  J IS COMPUTED BY EITHER THE USER SUPPLIED
C SUBROUTINE PEDERV, WHEN MITER = 1, OR BY FINITE DIFFERENCES,
C WHEN MITER = 2.  J IS STORED IN PW AND REPLACED BY P, USING
C CON = -H/EL(2).  THEN P IS SUBJECTED TO AN LU DECOMPOSITION
C FOR LATER SOLUTION OF LINEAR ALGEBRAIC SYSTEMS WITH P AS THE
C COEFFICIENT MATRIX.
C
C IN ADDITION TO VARIABLES DESCRIBED PREVIOUSLY, COMMUNICATION
C WITH PSET USES THE FOLLOWING..
C   EPSJ = SQRT(UROUND), USED IN THE NUMERICAL JACOBIAN INCREMENTS.
C   NSQ  = NO**2.
C----------------------------------------------------------------------
C CAUTION:  NOT ALL EPCOM1 VARIABLES ARE USED INTHIS SUBROUTINE.
C----------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE PSET.
C----------------------------------------------------------------------
      PARAMETER ( NMAX =120 )
      PARAMETER ( NMAXSQ = NMAX*NMAX )
      INTEGER IER, MITER, NO
      INTEGER IPIV, JSTART, KFLAG, MF, N, NSQ
      INTEGER I, J, J1
      REAL CON, Y
      REAL EPS, EPSJ, H, HMAX, HMIN, PW, SAVE1, SAVE2,
     1     SS, T, UROUND, YMAX
      REAL D, R, R0, YJ
      REAL ONE, REP, ZERO
C* Multiple Declaration of IER, T, N fixed - DRW
      DIMENSION Y(NO,1)
C
      COMMON /EPCOM1/ T,H,HMIN,HMAX,EPS,SS,UROUND,N,MF,KFLAG,JSTART
      COMMON /EPCOM2/ YMAX(NMAX)
      COMMON /EPCOM4/ SAVE1(NMAX)
      COMMON /EPCOM5/ SAVE2(NMAX)
      COMMON /EPCOM6/ PW(NMAXSQ)
      COMMON /EPCOM7/ IPIV(NMAX)
      COMMON /EPCOM8/ EPSJ,NSQ
      DATA ONE /1.0E0/, REP /1.0E-3/, ZERO /0.0E0/
      IF (MITER .EQ. 2) GOTO 20
C IF MITER = 1, CALL PEDERV AND MULTIPLY BY A SCALAR.  -----------------
      CALL PEDERV (N, T, Y, PW, NO)
      DO 10 I = 1,NSQ
 10     PW(I) = PW(I)*CON
      GOTO 60
C IF MITER = 2, MAKE N CALLS TO DIFFUN TO APPROXIMATE J. ---------------
 20   D = ZERO
      DO 30 I = 1,N
 30     D = D + SAVE2(I)**2
      R0 = ABS(H)*SQRT(D)*UROUND/REP
      J1 = 0
      DO 50 J = 1,N
        YJ = Y(J,1)
        R = EPSJ*YMAX(J)
        R = AMAX1(R,R0)
        Y(J,1) = Y(J,1) + R
        D = CON/R
        CALL DIFFUN (N, T, Y, SAVE1)
        DO 40 I = 1,N
 40       PW(I+J1) = (SAVE1(I) - SAVE2(I))*D
        Y(J,1) = YJ
        J1 = J1 + NO
 50     CONTINUE
C ADD ON THE IDENTITY MATRIX.  ----------------------------------------
 60   J = 1
      DO 70 I = 1,N
        PW(J) = PW(J) + ONE
 70     J = J + (NO + 1)
C GET LU DECOMPOSITION OF P. ------------------------------------------
      CALL DEC (N, NO, PW, IPIV, IER)
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE DEC (N, NDIM, A, IP, IER)
C---------------------------------------------------------------------
C MATRIX TRIANGULARIZATION BY GAUSSIAN ELIMINATION.
C INPUT..
C    N = ORDER OF MATRIX.
C    NDIM = DECLARED DIMENSION OF ARRAY  A .
C    A = MATRIX TO BE TRIANGULARIZED.
C OUTPUT..
C    A(I,J), I.LE.J = UPPER TRIANGULAR FACTOR, U .
C    A(I,J), I.GT.J = MULTIPLIERS = LOWER TRIANGULAR FACTOR, I - L.
C    IP(K), K.LT.N = INDEX OF K-TH PIVOT ROW.
C    IP(N) = (-1)**(NUMBER OF INTERCHANGES) OR O .
C    IER = 0 IF A NONSINGULAR, OR K IF A FOUND TO BE
```

```
C          SINGULAR AT STAGE K.
C USE  SOL  TO OBTAIN SOLUTION OF LINEAR SYSTEM.
C DETERM(A) = IP(N)*A(1,1)*A(2,2)*...*A(N,N).
C IF IP(N)=0, A IS SINGULAR, SOL WILL DIVIDE BY ZERO.
C INTERCHANGES FINISHED IN U , ONLY PARTLY IN L .
C
C REFERENCE..
C C. B. MOLER, ALGORITHM 423, LINEAR EQUATION SOLVER,
C COMM. ASSOC. COMPUT. MACH., 15 (1972), P. 274.
C-----------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE DEC.
C-----------------------------------------------------------------------
      INTEGER IER, IP, N, NDIM
      INTEGER I, J, K, KP1, M, NM1
      REAL A
      REAL T
      REAL ONE, ZERO
      DIMENSION A(NDIM,N),IP(N)
      DATA ONE /1.0E0/, ZERO /0.0E0/
      IER = 0
      IP(N) = 1
      IF (N .EQ. 1) GOTO 70
      NM1 = N - 1
      DO 60 K = 1,NM1
        KP1 = K + 1
        M = K
        DO 10 I = KP1,N
 10       IF (ABS(A(I,K)) .GT. ABS(A(M,K))) M = I
        IP(K) = M
        T = A(M,K)
        IF (M .EQ. K) GOTO 20
        IP(N) = -IP(N)
        A(M,K) = A(K,K)
        A(K,K) = T
 20     IF (T .EQ. ZERO) GOTO 80
        T = ONE/T
        DO 30 I = KP1,N
 30       A(I,K) = -A(I,K)*T
        DO 50 J = KP1,N
          T = A(M,J)
          A(M,J) = A(K,J)
          A(K,J) = T
          IF (T .EQ. ZERO) GOTO 50
          DO 40 I = KP1,N
 40         A(I,J) = A(I,J) + A(I,K)*T
 50       CONTINUE
 60     CONTINUE
 70   K = N
      IF (A(N,N) .EQ. ZERO) GOTO 80
      RETURN
 80   IER = K
      IP(N) = 0
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE SOL (N, NDIM, A, B, IP)
C---------------------------------------------------------------------
C SOLUTION OF LINEAR SYSTEM, A*X = B .
C INPUT..
C   N = ORDER OF MATRIX.
C   NDIM = DECLARED DIMENSION OF ARRAY  A .
C   A = TRIANGULARIZED MATRIX OBTAINED FROM DEC.
C   B = RIGHT HAND SIDE VECTOR.
C   IP = PIVOT VECTOR OBTAINED FROM DEC.
C DO NOT USE IF DEC HAS SET IER .NE. 0.
C OUTPUT..
C   B = SOLUTION VECTOR, X .
C---------------------------------------------------------------------
C$ THIS IS THE SINGLE PRECISION VERSION OF SUBROUTINE SOL.
C---------------------------------------------------------------------
      INTEGER IP, N, NDIM
      INTEGER I, K, KB, KM1, KP1, M, NM1
      REAL A, B
      REAL T
      DIMENSION A(NDIM, N), B(N), IP(N)
C
      IF (N .EQ. 1) GOTO 50
      NM1 = N - 1
      DO 20 K = 1,NM1
        KP1 = K + 1
        M = IP(K)
        T = B(M)
        B(M) = B(K)
        B(K) = T
        DO 10 I = KP1,N
 10       B(I) = B(I) + A(I,K)*T
 20     CONTINUE
      DO 40 KB = 1,NM1
        KM1 = N - KB
        K = KM1 + 1
        B(K) = B(K)/A(K,K)
        T = -B(K)
        DO 30 I = 1,KM1
 30       B(I) = B(I) + A(I,K)*T
 40     CONTINUE
 50   B(1) = B(1)/A(1,1)
      RETURN
      END
C
C=====================================================================
C
      BLOCK DATA
      COMMON /EPCOM9/ HUSED,NQUSED,NSTEP,NFE,NJE
      COMMON /EPCO99/ NCSTEP,NCFE,NCJE
      COMMON /EPCOMR/ NRMIN,NRMAX
      COMMON /EPCOMY/ YMIN,HMAXMX
      DATA HUSED,NQUSED,NSTEP,NFE,NJE / 0.,0,0,0,0 /
      DATA NCSTEP,NCFE,NCJE / 0,0,0 /
      DATA NRMIN,NRMAX / 1,500 /
      DATA YMIN,HMAXMX / 1.E-20,1.E6 /
      END
```

APPENDIX C:

LISTINGS OF DATA ACQUISITION CODES

The following pages contain source listings for the programs used to acquire smog chamber data on the PDP-11/03 RT-11 system during the toluene-NOx aerosol experiments. The programs consist of the following:
(1) DO2EAA, used to control and continuously sample two EAAs;
(2) DOEAA, used to control and continuously sample one EAA;
(3) SAVEAA, used to compress EAA data record from the above;
(4) VTEAA, used to display EAA size evolution on VT100 terminal;
(5) SAVOPC, used to conveniently enter OPC data off paper tape; and
(6) SAVTOL, used to enter toluene data from recorder tracings.

In addition, there is comprehensive documentation of my RTLIB package of over 120 subroutines and functions. These routines, written in Fortran and assembly language, enable the user who knows Fortran to take advantage of the capabilities of the lab RT-11 systems for A/D data sampling, D/A control, EAA control, timed sampling, video display and plotting, and some runtime statistics on the data. (The source listing of the many files comprising RTLIB is not included herein due to space and to the consideration that most users would more be interested in using it than changing it.) Extra documentation on the interconversion of various RT-11 formats for time is provided. Finally, there is documentation on the RT-11 system and source diskettes available for general use (telling where to find what).

Two other large data acquisition programs (listings of which are not included here because the programs did not figure in the toluene-NOx aerosol experiments, and to conserve space) are ASAP and WATCH. My modified ASAP program was used by Joe Leone et al. to do multichannel data sampling during the toluene chemistry experiments of 1983. It was not used for aerosol sampling because its once per channel A/D sampling of the EAA did not yield sufficiently high quality data from that instrument, and the ASAP program, half in assembly language, was difficult to modify. The WATCH program was to replace ASAP and include all the features of DO2EAA; unfortunately, by the time it had achieved continuous sampling of the normal analog instruments and a nice video display and command parser, it was pushing the capabilities of memory and the overlay handler. Nevertheless, if EAA control is not necessary, WATCH offers far more features than any other data acquisition package we have available on the RT-11. Also, WATCH is based on the RTLIB routines for easy modification.

```
      PROGRAM DO2EAA          ! Dual EAA sampling program
C
C   CONTINUOUS DUAL EAA SAMPLING & DISPLAY       ! 16-Sep-85 Version by DRW
C
C=>###  ###.### ####.#  ###.###-##.##E+00  ###.### ####.#  #####.###-##.###E+
C
      REAL*4 WAIT(10)         ! EAA Channel Wait Times in Seconds
      REAL*4 V1(10),V2(10)              ! EAA Channel Voltages (average)
      REAL*4 VSD1(10),VSD2(10)          ! EAA Standard Deviations in Volts
      REAL*4 VSL1(10),VSL2(10)          ! EAA Regressions, Volts/Second
      INTEGER NS1(10),NS2(10)           ! Number of EAA Readings Taken on A/D
      REAL*4 PN1(10),PV1(10)  ! EAA Distribution, #/cc & cu.um./cc
      REAL*4 PN2(10),PV2(10)  ! ditto, 2nd EAA
      BYTE FILE(20)           ! Output Data File Name
      BYTE TSTR(12)           ! Time String
      BYTE STRING(81)
      INTEGER*4 TICKS
      BYTE ASK,NORMAL,CH,EOL
      LOGICAL*1 NOTICK
C
      EXTERNAL EAACOM
C
      COMMON /ADCALO/ VCON
      COMMON /EAA1/ V1,VSD1,VSL1,NS1            ! Arrays (10) for first EAA
      COMMON /EAA2/ V2,VSD2,VSL2,NS2            ! Arrays (10) for second EAA
C
      COMMON /KILL/ KILL                        ! Not yet used
      COMMON /EAACY/ NREP,ISKIP,REMODE          ! REMODE not used yet
      COMMON /EAACH/ KEAA1,KEAA2,IGAIN
C
      DATA WAIT /30.,20.,16.,10.,6.,6.,6.,4.,4.,4./
C     DATA WAIT / 10 * 3. /             ! Dummy for Testing Only
      DATA KEAA1,KEAA2 / 14,15 /
      DATA IGAIN / 2 /
      DATA EOL / 0 /          ! EOL = End Of Line = String Termination Byte
      DATA NOTICK / .TRUE. /  ! Omit Ticks from Times
      DATA KILL / 0 /         ! Used to stop EAA cycling
C
C     Timing for POOH and DUAL smog chamber experiments (Tcycle=163.3 sec):
C     NREP = 300 ; ISKIP=1 ; WAIT=30,20,16,10,6,6,6,4,4,4
C
      NREP =  300             ! 300 Readings per Channel Sample
      ISKIP = 1               ! Skip 1 Tick between Readings
C
C***           SETUP QUESTIONS
C
      CALL VINIT
      CALL VSCROL(3,23)       ! Top and Bottom Line Will Not Scroll
      CALL VSTR('VT PROGRAM for Dual EAA SAMPLING',1,11,'B',1)
      CALL VSTR(' by ',0,0,' ',1)
      CALL VSTR('DRW',0,0,'BF',1)
      CALL VSTR('  (Sep-85 Version)',0,0,' ',0)
```

```
        CALL SETEAA
        CALL ISLEEP(0,0,2,0)
        CALL VPUT(3,1)
        CALL ASKADC                 ! Check A/D Scale
        CALL ASKYN('Single cycle mode, user begins each cycle?','N',CH)
 1      FORMAT(A1)
        TYPE 14
14      FORMAT(/'$Enter # of cycles to run (0=No limit): ')
        ACCEPT 2, NCYCLE
        IC = 0                      ! Current Cycle Number
 2      FORMAT(I)
        TYPE 18
18      FORMAT(/'$EAA Data Storage -- ')
        CALL SCOPY('DATA.E2R',FILE)       ! Default
        CALL ASKFIL(2,FILE)               ! Ask for Filename & Open on 2
        WRITE(2,21)
21      FORMAT(' Using D02EAA Sampling Program as of 16-Sep-85')
        CALL ASKYN('Shall we use the default EAA timing?','Y',NORMAL)
        IF (NORMAL.NE.'N') GOTO 30
        TYPE 24
24      FORMAT(/' You may now change the EAA Settling Times.'/)
        CALL CHEAAT(WAIT)
        TYPE 26, ISKIP
26      FORMAT(/' Currently A/D read occurs each',I3,' clock ticks ',
        1 '(60 ticks/sec).')
        CALL ASKI('Number of Ticks between A/D Reads:',1,ISKIP)
27      TYPE 28, NREP
28      FORMAT(/' Currently each channel voltage is based on',I5,
        1  ' A/D reads.')
        CALL ASKI('Readings per average:',1,NREP)
        IF (NREP.LE.0) GOTO 27
30      WRITE(2,33) (WAIT(K),K=1,10)
33      FORMAT(' Settling Times: ',10F6.1)
        WRITE(2,34) ISKIP,NREP
34      FORMAT(' ISKIP =',I4,' ticks',5X,'NREP =',I5,' A/D readings')
        TYPE 35, KEAA1,KEAA2
35      FORMAT(/'$Are the two EAAs connected to A/D channels',I3,
        1  ' and',I3)
        CALL ASKYN('?','Y',ASK)
        IF (ASK.NE.'N') GOTO 36
        TYPE 32, 'first '
        CALL ASKI(' ',1,KEAA1)
32      FORMAT(/'$Enter A/D channel for ',A6,' EAA (0-15): ')
        TYPE 32, 'second'
        CALL ASKI(' ',1,KEAA2)
36      CALL ISLEEP(0,0,2,0)
        CALL VCLEAR
        CALL VSCROL(21,23)        ! Only Short Scrolling Region
        FILE(20)=0
        CALL VPUT(1,63)
        CALL VPRINT(FILE)
        CALL VSETG1
        CALL VSTR('~DUAL EAA SAMPLING PROGRAM~',1,34,'BR',0)
```

```
       CALL VSETGO
       CALL VPUT(4,1)
C   FORMAT(A3,I3,2X,F7.3,1X,F6.1,2X,F7.3,1PE10.2,2X,F7.3,1X,F6.1,2X,F9.3,E10.2)
       CALL VPRINT('   Chan   Volts  mV sd    Volts    #/cc  ')
       CALL VPRINT('       Volts  mV sd    Volts   #/cc')
       CALL ISLEEP(0,0,1,0)
       CALL VPUT(6,1)
       DO 39 K=1,10
         ICHAN=K+1
         ENCODE(6,38,STRING) ICHAN,EOL
38       FORMAT(4X,I2,A1)
         CALL PRINT(STRING)
39     CONTINUE
C
C      ENCODE(78,37,STRING) (J,J=2,11),EOL
C37    FORMAT(1X,10I7,A1)
C      CALL VPUT(17,1)
C      CALL PRINT(STRING)
C      CALL PRINT('V1')
C      CALL PRINT('V2')
C
       CALL EAACOM('H')
C
C***            READY TO CYCLE EAA
C
40     IF (CH.NE.'Y') GOTO 50
       CALL VPUT(24,1)
       CALL VPRINT('Hit RETURN to Begin Sampling (or E to END) : ')
       CALL SPECIN(ASK,0)
       CALL VPUT(24,1)
       CALL VHLINE(24,'  ',1,75)
       IF (ASK.EQ.'S') CALL STEP
       IF (ASK.EQ.'E') GOTO 990
50     CALL RESET                ! Intialize EAA Sampling Routine
       IC=IC+1                   ! Cycle Count
       ENCODE(10,55,STRING) IC,EOL
55     FORMAT('Cycle #',I3,A1)
       CALL VPUT(1,22)
       CALL VPRINT(STRING)
       CALL STEP                 ! Step to Starting Channel, now #2
       DO 150 KK=1,10            ! Loop Through The Channels (#2 to #11)
        K=KK                     ! Avoid Compiler Warning
        ICHAN=K+1
        IF (ICHAN.LT.10) ENCODE(2,51,ECHAN) ICHAN,EOL
        IF (ICHAN.GE.10) ENCODE(2,52,ECHAN) ICHAN,EOL
51      FORMAT(I1,1X,A1)
52      FORMAT(I2,A1)
        LINEX=ICHAN+4
        CALL VSTR('==>',LINEX,1,'BF',0)         ! Arrow denotes EAA channel
        CALL CLOCK(WAIT(K),1,0,-2,1,EAACOM)     ! Waits; EAACOM if char input
        CALL VPUT(1,9)
        CALL VPRINT('  Read # ')
        CALL VPRINT(ECHAN)
```

```
         CALL S2EAA(K)                              ! Sample Channel
         CALL VSTR('    ',LINEX,1,' ',0)           ! Clear Arrow
         CALL VPUT(LINEX,1)
         CALL STEP                ! Step to . . .
         KCOL=7*K-3
         ENCODE(15,70,STRING) V1(K),1000.*VSD1(K),EOL
70       FORMAT(F7.3,1X,F6.1,A1)
         CALL VSTR(STRING,LINEX,9,' ',0)
         ENCODE(15,70,STRING) V2(K),1000.*VSD2(K),EOL
         CALL VSTR(STRING,LINEX,45,' ',2)
150     CONTINUE                  ! Next Channel
C
C              DONE WITH A EAA CYCLE
C
        CALL GTIM(TICKS)
        HRS=HRTINT(TICKS)
        CALL CVHRST(HRS,TSTR)
        IF (NOTICK) TSTR(9)=0
        CALL VPUT(17,55)
        CALL VPRINT(' End of Cycle at  ')
        CALL VPRINT(TSTR)
        WRITE(2,800) (TSTR(I),I=1,8)
800     FORMAT(1X,'End of Cycle Time is ',8A1,3X)
        CALL EAASIZ(V1,PN1,PV1,TN1,TV1,TS1,TD1) ! Calculate Distribution
        CALL EAASIZ(V2,PN2,PV2,TN2,TV2,TS2,TD2)
        CALL VPUT(5,1)
        DO 850 KK=1,10
         K=KK                     ! Avoid Compiler Warning
         LINEX=K+5
C  FORMAT(A3,I3,2X,F7.3,1X,F6.1,2X,F7.3,1PE10.2,2X,F7.3,1X,F6.1,2X,F9.3,E10.2)
         ENCODE(26,805,STRING) 1000.*VSD1(K),V1(K),PN1(K),EOL
         IF (PN1(K).EQ.0.) CALL SCOPY(' 0        ',STRING(17))
         CALL VHLINE(LINEX,'  ',8,16)
         CALL VSTR(STRING,LINEX,17,' ',1)
         ENCODE(26,805,STRING) 1000.*VSD2(K),V2(K),PN2(K),EOL
         IF (PN2(K).EQ.0.) CALL SCOPY(' 0        ',STRING(17))
         CALL VHLINE(LINEX,'  ',47,53)
         CALL VSTR(STRING,LINEX,53,' ',2)
805      FORMAT(F6.1,2X,F7.3,1X,1PE9.2,A1)
C       FORMAT(A3,I3,F10.3,F6.1,F9.3,1PE10.2,4X,0PF7.3,F6.1,F9.3,1PE10.2)
         ENCODE(77,810,STRING) K+1,V1(K),1000.*VSD1(K),PN1(K),
        1  V2(K),1000.*VSD2(K),PN2(K),NS1(K),EOL
810     FORMAT(I3,2X,F6.4,2X,F6.1,2X,1PE10.3,4X,
        1  0PF6.4,2X,F6.1,2X,1PE10.3,2X,I7,A1)
         L=LEN(STRING)
         WRITE(2,811) (STRING(I),I=1,L)
811      FORMAT(77A1)
C        CALL VSETG1
C        CALL VPUT(LINEX,15)
C        CALL VPRINT('g')
C        CALL VPUT(LINEX,51)
C        CALL VPRINT('g')
C        CALL VSETG0
```

```
850     CONTINUE
        CALL VPUT(19,1)
        ENCODE(75,880,STRING) 1,TN1,TN1-PN1(1),TV1,EOL
        CALL PRINT(STRING)
        ENCODE(75,880,STRING) 2,TN2,TN2-PN2(1),TV2,EOL
        CALL PRINT(STRING)
880     FORMAT(' By EAA ',I1,', Number=',1PE10.3,' (',1PE10.3,
        1 ') /cc',2X,'Volume=',1PE10.3,' cu.um./cu.m.',A1)
C       CALL VHLINE(18,'  ',1,79)
C       CALL VHLINE(19,'  ',1,79)
C
        IF (KILL.EQ.-1) GOTO 40
        IF (KILL.EQ.1) GOTO 990
        IF (NCYCLE.GT.0 .AND. IC.GE.NCYCLE) GOTO 990
        GOTO 40
990     CALL VSCROL(1,24)
        CALL VPUT(21,1)
        STOP 'EAA SAMPLING COMPLETE'
        END


        SUBROUTINE S2EAA(K)              ! Sample Channel on Both EAAs
C
C       This subroutine merely reads the EAAs on the A/D with
C       information passed in COMMONs.  No EAA control here.
C
        COMMON /EAA1/ V1(10),VSD1(10),VSL1(10),NS1(10)
        COMMON /EAA2/ V2(10),VSD2(10),VSL2(10),NS2(10)
        COMMON /AWAKE/ AWAKE             ! Timed Completion Flag (Optional)
        COMMON /EAACY/ NREP,ISKIP,REMODE
        COMMON /EAACH/ KEAA1,KEAA2       ! A/D Voltage Unit
        DATA AWAKE / 0. /                ! Completion Flag Off
        NS = 0                           ! Number of Readings
        VSUM = 0.                        ! Voltage Sum
        V2SUM = 0.                       ! Voltage Sum Squared
        VISUM = 0.                       ! Voltage-Time Product Sum
        WSUM = 0.                        ! Voltage Sum
        W2SUM = 0.                       ! Voltage Sum Squared
        WISUM = 0.                       ! Voltage-Time Product Sum
        CALL GTIM(START)                 ! Starting Time
100     CALL VREAD(KEAA1,VOLTS)          ! Read A/D for first EAA
        CALL VREAD(KEAA2,WOLTS)          ! Read A/D for second EAA
        NS=NS+1                          ! Maintain statistical summations
        VSUM=VSUM+VOLTS
        V2SUM=V2SUM+VOLTS*VOLTS
        VISUM=VISUM+FLOAT(NS)*VOLTS
        WSUM=WSUM+WOLTS
        W2SUM=W2SUM+WOLTS*WOLTS
        WISUM=WISUM+FLOAT(NS)*WOLTS
        IF (AWAKE.EQ.1. .OR. NS.GE.NREP) GOTO 200       ! Escape loop
        CALL ISLEEP(0,0,0,ISKIP)         ! Delay
        GOTO 100                         ! Loop to Read Again
200     CALL GTIM(FINISH)
```

```
      RSEC = ELAPSE(START,FINISH)
      CALL STATS(NS,VSUM,V2SUM,VISUM,V1(K),VSD1(K),VSL1(K))
      CALL STATS(NS,WSUM,W2SUM,WISUM,V2(K),VSD2(K),VSL2(K))
      VSL1(K) = VSL1(K) * FLOAT(NS) / RSEC          ! Regression Rates,
      VSL2(K) = VSL2(K) * FLOAT(NS) / RSEC          !  Volts/Sec
      NS1(K)=NS
      NS2(K)=NS
      AWAKE=0.                          ! Reset Timed Completion Flag
500   RETURN
      END
      PROGRAM DOEAA           ! EAA sampling program, Version 6
C
C   CONTINUOUS EAA SAMPLING & DISPLAY PROGRAM   ! Written by DRW Jan-85
C                                                ! 9-Jan-86 Revision
C
C   LINKs with RTLIB to use:
C       Assembly language routines for EAA control.
C       Simple EAA data reduction routines.
C       Video display routines.
C       Timing and Time conversion.
C
      REAL*4 WAIT(10)          ! EAA Channel Wait Times in Seconds
      REAL*4 VSIG(10)          ! EAA Voltages (Average)
      REAL*4 VSD(10)           ! EAA Standard Deviation in Voltage
      REAL*4 VREG(10)          ! EAA Time Regression in Voltage
      REAL*4 PN(10),PV(10)     ! EAA Distribution, #/cu.m. & cu.um./cu.m.
      INTEGER*2 NS(10)         ! Number of EAASAM's Taken
      BYTE FILE(20)            ! Output Data File Name
      BYTE TSTR(12)            ! Time String
      BYTE STRING(81),INCHAR
      INTEGER*4 TICKS
      LOGICAL*1 BRIEF
      BYTE ASK,NORMAL,ECHAN(3),CLTIME(12),EOL
C
      EXTERNAL EAACOM
C
      COMMON /EAA1/ VSIG,VSD,VREG,NS          ! All dimensioned (10)
      COMMON /KILL/ KILL                      ! Now used
      COMMON /EAACY/ NREP,ISKIP,REMODE        ! REMODE not used yet
      COMMON /EAACH/ KEAA                     ! A/D Channel of EAA
      COMMON /CYTIM/ LHOLD,LHR,LMI,LSE,LTI,AREAL(2)   ! Cycle Time
      COMMON /CLOCK1/ CTINT,CLTIME            ! CLOCK Time of Day
      COMMON /TINPUT/ IOFLAG,IOMODE,INCHAR
C
      DATA WAIT /30.,20.,16.,10.,6.,6.,6.,4.,4.,4./
C     DATA WAIT / 10 * 3. /    ! Dummy for Quick Debugging Only
      DATA EOL  / 0 /          ! EOL = End of Line = String Termination
      DATA ECHAN / 3*0 /       ! Character String
      DATA CYSEC / 0. /        ! Default is no minimum cycle time
      DATA LHOLD / 0 /         ! Default is no cycle HOLD
C
      KEAA = 14                ! EAA is read on A/D Channel 14
      NREP =  300              ! 300 Readings per EAA Channel Sample
      ISKIP = 1                ! Skip 1 Tick between Readings
C
C***            SETUP QUESTIONS
C
      CALL VINIT
      CALL VSCROL(3,24)        ! Top Two Lines Will Not Scroll
      CALL VSTR('VT PROGRAM for EAA SAMPLING',1,11,'B',1)
      CALL VSTR('by',1,40,' ',1)
```

```
        CALL VSTR('DRW',1,43,'BF',1)
        CALL VSTR('(Jan-86 Version)',1,47,' ',1)
        CALL SETEAA                ! Set up for EAA analysis
        CALL ISLEEP(0,0,2,0)       ! Hold Display
        CALL VPUT(3,1)
        CALL ASKADC                ! Check A/D Range (/ADCALO/ VCON set)
        TYPE 1
 1      FORMAT(' ')
        CALL ASKYN('Single Cycle Mode?','N',CHAR)
        TYPE 14
14      FORMAT(/'$Enter # of cycles to run [0=no limit] : ')
        ACCEPT 2, NCYCLE
 2      FORMAT(I)
        TYPE 1
        CALL ASKYN('Use Compact Storage?','N',ASK)
        BRIEF=.FALSE.
        IF (ASK.EQ.'Y') BRIEF=.TRUE.
        TYPE 18
18      FORMAT(/'$EAA Data Storage -- ')
        CALL SCOPY('DATA.ER',FILE)
        IF (BRIEF) CALL SCOPY('DATA.EA',FILE)
        CALL ASKFIL(2,FILE)                ! Ask for Filename, Open on 2
        TYPE 1
        CALL ASKYN('Shall we use the default EAA timing?','Y',NORMAL)
        IF (NORMAL.EQ.'Y') GOTO 30
        TYPE 24
24      FORMAT(/' You may now change the EAA Settling Times.'/)
        CALL CHEAAT(WAIT)
        TYPE 26, ISKIP
26      FORMAT(/' Currently A/D read occurs each',I3,' clock ticks ',
        1 '(60 ticks/sec).')
        CALL ASKI('Number of Ticks between A/D Reads:',1,ISKIP)
27      TYPE 28, NREP
28      FORMAT(/' Currently each channel voltage is based on',I5,
        1  ' A/D reads.')
        CALL ASKI('Readings per Average:',1,NREP)
        IF (NREP.LE.0) GOTO 27
30      IF (.NOT.BRIEF) WRITE(2,31)
31      FORMAT(' Using DOEAA Sampling Program as of 9-Jan-86')
        IF (.NOT.BRIEF) WRITE(2,32) (WAIT(I),I=1,10)
32      FORMAT(' Settling Times: ',10F6.1)
        IF (.NOT.BRIEF) WRITE(2,33) ISKIP,NREP
33      FORMAT(' ISKIP =',I4,' ticks',5X,'NREP =',I5,' A/D readings')
        IF (BRIEF) WRITE(2,*) 2,11,0,0          ! ICHAN,LCHAN,KDIF,MAXSET
        TYPE 1
        CALL ASKR('Enter Minimum Cycle Time (Seconds):',1,CYSEC)
        CYHR=CYSEC/3600.
        CALL CVHRT4(CYHR,LHR)              ! Convert to INTEGER Time Foursome
C
        TYPE 1
        CALL ASKI('ENTER A/D Channel for EAA',1,KEAA)
        TYPE 11, KEAA
11      FORMAT(/T5,'Please Make Sure EAA Is Connected, Using A/D',
      1  ' Port',I3/)
      IC = 0                     ! Cycle #0 in progress
      CALL ISLEEP(0,0,4,0)
      CALL VCLEAR
      CALL VSCROL(21,23)         ! Only Bottom Lines will Scroll
      CALL VPUT(1,60)
      FILE(20)=0
      CALL VPRINT(FILE)
      CALL VSETG1
      CALL VSTR('~EAA SAMPLING PROGRAM~',1,35,'BR',0)
      CALL VSETG0
      CALL ISLEEP(0,0,1,0)
      CALL VSTR('   EAA     Signal   St.Dev.   Slope ',3,1,' ',1)
      CALL VSTR('  Current   Number  Volume',0,0,' ',1)
      CALL VSTR('   Chan    Volts       mV      mV/sec',4,1,' ',1)
      CALL VSTR('    pA       #/cc    cu/cc',0,0,' ',1)
      CALL VPUT(6,1)
      DO 39 K=1,10                         ! Label EAA Channels
        ICHAN=K+1      ! Dummy Loop Index to Avoid Dumb Compiler Warning
        ENCODE(6,38,STRING) ICHAN,EOL
38      FORMAT(4X,I2,A1)
        CALL PRINT(STRING)
39    CONTINUE
C
C***           READY TO CYCLE EAA
C
40    IF (CHAR.NE.'Y' .AND. IC.GT.0) GOTO 50
      CALL VPUT(24,1)
      CALL VPRINT('Hit Any Key to Begin Sampling (or E to END) : ')
      CALL CLOCK(36000.,1,0,0,3)       ! Civilized waiting for user input
C     CALL SPECIN(INCHAR,0)            ! (Wait without time of day display)
C
C***           BEGIN CYCLE BY ACTIVATING CYCLE TIMER
C
      IF (CHAR.NE.'Y'.AND.CYSEC.GT.0.) CALL LCYCLE    ! Self-Scheduler
      CALL VPUT(24,1)
      CALL VHLINE(24,'  ',1,75)
      CALL EAACOM('H')
      IF (INCHAR.EQ.'E') GOTO 990
50    CALL RESET                 ! Intialize EAA Sampling Routine
      IF (CHAR.NE.'Y'.AND.CYSEC.GT.0.) LHOLD=1         ! Set Cycle Timer Flag
      IC=IC+1                    ! Cycle Count
      ENCODE(12,55,STRING) IC,EOL
55    FORMAT('Cycle #',I3,A1)
      CALL VSTR(STRING,1,22,' ',0)
      CALL STEP                  ! Step to Starting Channel, now #2
C
C***           CYCLE EAA
C
      DO 150 K=1,10              ! Loop Through The Channels (#2 to #11)
       ICHAN=K+1
       IF (ICHAN.LT.10) ENCODE(2,51,ECHAN) ICHAN,EOL
       IF (ICHAN.GE.10) ENCODE(2,52,ECHAN) ICHAN,EOL
```

```
 51         FORMAT(I1,1X,A1)
 52         FORMAT(I2,A1)
            LINEX=ICHAN+4
            CALL VSTR('==>',LINEX,1,'BF',0)         ! Arrow denotes EAA Channel
            CALL CLOCK(WAIT(K),1,0,-2,1,EAACOM)     ! Wait Time
            CALL VPUT(1,9)
            CALL VPRINT('  Read # ')
            CALL VPRINT(ECHAN)
            CALL SEAA(VSIG(K),VSD(K),VREG(K),NS(K)) ! Sample Channel
            CALL VPUT(LINEX,1)
          ENCODE(75,810,STRING) ICHAN,VSIG(K),1.E3*VSD(K),1.E3*VREG(K),EOL
            CALL VPRINT(STRING)
            CALL STEP                 ! Step to . . .
150       CONTINUE                    ! Next Channel
C
C                DONE WITH A EAA CYCLE
C
          CALL CLOCK(-1.,1,0,-2,0)                 ! Just Print Out Time of Day
          CALL VPUT(1,9)
          CALL VPRINT(' Cycle Done ')
          HOUR=HRTINT(CTINT)
          ENCODE(32,750,STRING) IC,(CLTIME(I),I=1,8),EOL
750       FORMAT(' Cycle #',I4,' Ended at ',8A1,A1)
          IF (.NOT.BRIEF) WRITE(2,760) (STRING(I),I=1,31)
760       FORMAT(80A1)
          CALL VSTR(STRING,18,38,' ',0)
          CALL EAASIZ(VSIG,PN,PV,TN,TV,TS,TD)      ! Calculate Distribution
          CALL VPUT(6,1)
          DO 850 K=1,10
          ENCODE(75,810,STRING) K+1,VSIG(K),1000.*VSD(K),1000.*VREG(K),EOL
810       FORMAT(4X,I2,3X,F9.4,1X,F7.1,1X,F7.1,A1)
            IF (.NOT.BRIEF) WRITE(2,811) (STRING(I),I=1,34)
811         FORMAT(75A1)
            CALL VPRINT(STRING)
            ENCODE(35,815,STRING) VSIG(K),PN(K),PV(K),EOL
            IF (PN(K).EQ.0.) CALL SCOPY('  0.        0.',STRING(11))
815         FORMAT(1X,0PF9.4,1P2E10.2,A1)
            CALL PRINT(STRING)
850       CONTINUE
C
C         BRIEF generates the new standard .EA current files; no st.dev. saved
C
          IF (BRIEF) WRITE(2,855) HOUR,(VSIG(K),K=1,10)
855       FORMAT(11F7.4)
C           CALL VSETG1
C           CALL VPUT(K+2,21)
C           CALL PRINT('g')
C           CALL VSETG0
          CALL VPUT(20,1)
          TN3=TN-PN(1)                      ! Chan 3 up (.01 um)
          TN4=TN3-PN(2)                     ! Chan 4 up (.18 um)
          ENCODE(75,880,STRING) TN,TN3,TN4,EOL
          CALL PRINT(STRING)
```

```
      ENCODE(75,885,STRING) TV,EOL
      CALL PRINT(STRING)
880   FORMAT(' Total Number is',1PE11.3,' ;',E11.3,' ;',E11.3,
      1  ,' /cc  (Chan 2+,3+,4+)',A1)
885   FORMAT(' Total Volume is',1PE11.3,' um**3/cc',A1)
C
      IF (KILL.EQ.-1) GOTO 40                  ! Don't Stop Now
      IF (KILL.EQ.1) GOTO 990                  ! Do Stop Now
      IF (NCYCLE.GT.0 .AND. IC.GE.NCYCLE) GOTO 990
      IF (LHOLD.EQ.1) CALL VSTR(' Timed Wait ',1,9,' ',0)
888   IF (LHOLD.EQ.1) GOTO 888         ! Wait for Cycle Timer Flag
      GOTO 40
990   CALL VSCROL(1,24)
      CALL VPUT(22,1)
      STOP 'EAA SAMPLING COMPLETE          '
      END


      SUBROUTINE SEAA(AV,SD,REG,NS)    ! Sample an EAA Channel
      COMMON /EAACH/ KEAA              ! A/D Channel of EAA
      COMMON /EAACY/ NREP,ISKIP,REMODE
      COMMON /AWAKE/ AWAKE             ! Completion Flag
      AWAKE = 0.                       ! Turn Off Stop Signal
      NS = 0                   ! Number of Readings
      VSUM = 0.                ! Voltage Sum
      V2SUM = 0.               ! Voltage Sum Squared
      VISUM = 0.               ! Voltage-Time Product Sum
      CALL GTIM(START)
100     CALL VREAD(KEAA,VOLTS)         ! Read EAA via A/D
        NS=NS+1
        VSUM=VSUM+VOLTS
        V2SUM=V2SUM+VOLTS*VOLTS
        VISUM=VISUM+FLOAT(NS)*VOLTS
        IF (AWAKE.EQ.1. .OR. NS.GE.NREP) GOTO 200
        CALL ISLEEP(0,0,0,ISKIP)
      GOTO 100
200   CALL GTIM(FINISH)
      RUNSEC = ELAPSE(START,FINISH)
      AV = 0.
      VAR = 0.
      SD = 0.
      REG = 0.
      XN = FLOAT(NS)
      IF (NS.EQ.0) GOTO 500    ! No Samples
      AV = VSUM / XN
      IF (NS.LE.1) GOTO 500    ! One Sample
      VAR = (V2SUM - VSUM * VSUM / XN) / (XN - 1.)
      IF (V2SUM.EQ.0.) GOTO 300
      IF (VAR/V2SUM.LT.-1.E-6) TYPE *,'Variance Seriously Negative'
      IF (VAR.LT.0.) VAR=0.
300   SD = SQRT (VAR)
      DEN = XN * (XN+1.) * (XN-1.) / 12.
      PS = 0.5 * VSUM * (XN+1.)
```

```
        REG = ( VISUM - PS ) / DEN       ! Per A/D Read Regression
        REG = REG * XN / RUNSEC          ! Regression Rate
500     RETURN
        END
C
C
        SUBROUTINE LCYCLE                        ! Timed Completion Routine
        EXTERNAL LCYCLE
        COMMON /CYTIM/ LHOLD,LHR,LMI,LSE,LTI,AREAL(2)    ! Cycle Time
        LHOLD=0                                  ! Cancel Hold
        CALL ITIMER(LHR,LMI,LSE,LTI,AREAL,29,LCYCLE)     ! Self-Schedules
        RETURN
        END
```

```
      PROGRAM SAVEAA          ! EAA RAW DATA SIMPLIFICATION
C
C     Designed to work with the DOEAA and DO2EAA data sets.
C     Written by DRW.  Revised 15-Nov-85
C     Transfers long ASCII .E2R files to ASCII voltage .EA1 and .EA2 files
C     These files are handled directly by the IBM AT programs THREATS, etc.
C
      REAL V1(10),V2(10)
      BYTE EFILE(20),OFILE1(20),OFILE2(20)
      BYTE STRING(81),TSTR(8),CC,MAYBE
        DATA CC / ' ' /       ! File Carriage Control Character
      LOGICAL*1 ERR,DOLIST
C
      CALL VINIT
      CALL VSTR('EAA DATA TRANSFER PROGRAM',1,20,'B',1)
      CALL ISLEEP(0,0,1,0)
      CALL VPRINT(' by ')
      CALL VSTR('DRW',0,0,'BF',0)
      CALL VSTR('   Nov 1985',0,0,' ',2)
      CALL ISLEEP(0,0,1,0)
 16   CALL VPUT(4,1)
      MODE=2
      CALL ASKI('EAA Mode (1=single, 2=dual) : ',1,MODE)
      IF (MODE.GT.2 .OR. MODE.LT.1) GOTO 16
      CALL VPUT(7,1)
      CALL VPRINT('EAA Input Data File ')
      IF (MODE.EQ.2) CALL VPRINT('[.E2R] - ')
      IF (MODE.EQ.1) CALL VPRINT('[.ER] - ')
      IF (MODE.EQ.2) CALL SCOPY('DATA.E2R',EFILE)
      IF (MODE.EQ.1) CALL SCOPY('DATA.ER',EFILE)
      CALL ASKNAM(EFILE)               ! Open input on LUN 2
      LDOT=INDEX(EFILE,'.')
      IF (LDOT.NE.0) GOTO 20
      IF (MODE.EQ.2) CALL CONCAT(EFILE,'.E2R',EFILE)
      IF (MODE.EQ.1) CALL CONCAT(EFILE,'.ER',EFILE)
 20   CALL ASSIGN(2,EFILE)             ! Open output .EA1 on LUN 3
      CALL VPUT(10,1)
      LDOT=INDEX(EFILE,'.')
      CALL SCOPY(EFILE,OFILE1,LDOT)
      IF (LDOT.EQ.0) CALL SCOPY('DATA.',OFILE1)
      CALL CONCAT(OFILE1,'EA1',OFILE1)
      CALL VPRINT('EAA Output Data File 1 [.EA1] - ')
      CALL ASKNAM(OFILE1)
      CALL TRIM(OFILE1)
      LO=LEN(OFILE1)
      LDOT=INDEX(OFILE1,'.')
      IF (LO.EQ.0) CALL SCOPY('NL:',OFILE2)
      IF (LDOT.EQ.0.AND.LO.NE.0) CALL CONCAT(OFILE1,'.EA1',OFILE1)
      CALL ASSIGN(3,OFILE1)            ! Open output .EA1 on LUN 3
      IF (MODE.EQ.1) GOTO 25
      CALL VPUT(12,1)
      LDOT=INDEX(OFILE1,'.')
      CALL SCOPY(OFILE1,OFILE2,LDOT)
      CALL CONCAT(OFILE2,'EA2',OFILE2)
      CALL VPRINT('EAA Output Data File 2 [.EA2] - ')
      CALL ASKNAM(OFILE2)
      CALL TRIM(OFILE2)
      LO=LEN(OFILE2)
      LDOT=INDEX(OFILE2,'.')
      IF (LO.EQ.0) CALL SCOPY('NL:',OFILE2)
      IF (LDOT.EQ.0.AND.LO.NE.0) CALL CONCAT(OFILE2,'.EA2',OFILE2)
      CALL ASSIGN(4,OFILE2)            ! Open output .EA2 on LUN 4
 25   CALL VPUT(15,1)
      CALL ASKYN('Do you want terminal listing?','N',MAYBE)
      DOLIST=.TRUE.
      IF (MAYBE.EQ.'N') DOLIST=.FALSE.
      IF (.NOT.DOLIST) GOTO 26
      CALL ISLEEP(0,0,1,0)
C
      CALL VCLEAR
      CALL V132C
C
 11   FORMAT(A80)
 26   CALL GETSTR(2,STRING,80,ERR)
      CALL PRINT(STRING)
      CALL GETSTR(2,STRING,80,ERR)
      CALL PRINT(STRING)
      CALL GETSTR(2,STRING,80,ERR)
      CALL PRINT(STRING)
C      CALL PUTSTR(6,STRING,CC,ERR)   ! Alternate, goes to LP: normally
C
      WRITE(3,*) 2,11,0,0
      IF (MODE.EQ.2) WRITE(4,*) 2,11,0,0
      IF (.NOT.DOLIST) GOTO 115
      CALL VSCROL(2,24)
      CALL VPUT(1,1)
      CALL VBOLD
      TYPE 100, (I,I=2,11)
      CALL VOFF
100   FORMAT('   EAA  ',10I10)
C
C               READ IN ENDING TIME AND CHANNEL VOLTAGES
C
115   READ(2,118,END=500,ERR=500) TSTR
118   FORMAT(22X,8A1)
      KOUNT = KOUNT + 1
      HOUR=HRSST(TSTR)
      DO 125 K=1,10
        IF (MODE.EQ.1) READ(2,120,END=450) V1(K)
120     FORMAT(11X,F6.4)
        IF (MODE.EQ.2) READ(2,121,END=450) V1(K),V2(K)
121     FORMAT(4X,F7.4,23X,F7.4)
125   CONTINUE
C
      IF (.NOT.DOLIST) GOTO 139
```

```
        TYPE 135, (TSTR(I),I=1,8),(V1(I),I=1,10)
        IF (MODE.EQ.2) TYPE 135, (TSTR(I),I=1,8),(V2(I),I=1,10)
135     FORMAT(1X,8A1,3X,10(F8.4,2X))
139     WRITE(3,140) HOUR,(V1(I),I=1,10)
        IF (MODE.EQ.2) WRITE(4,140) HOUR,(V2(I),I=1,10)
140     FORMAT(11F7.4)
C
        GOTO 115                             ! LOOP
C
450     TYPE *, 'UNEXPECTED END OF FILE'
500     CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        IF (MODE.EQ.2) CLOSE (UNIT=4)
        TYPE *,' '
        IF (MAYBE.EQ.'Y') CALL V80C
        STOP 'SavEAA Program Finished'
        END
```

```
      PROGRAM VTEAA             ! EAA DATA GRAPHING PROGRAM
C
C     Designed to work with the .EA# EAA data sets from SAVEAA
C     Revised Nov-85 by DRW.  Designed for VT100 or Printouts.
C     With option to plot Concentrations on Screen.
C
      COMMON /EAA/ EAADP(10),EAADPM(9)
      REAL T(200)               ! Time Array (hours)
      REAL Y(200,13)            ! Particle Number
      REAL YMAX(13)
      REAL V(10),PN(9),PV(9)
      INTEGER EFLAG
      BYTE EFILE(20),OFILE(20),STRING(81),CC,MAYBE,CHAR
        DATA CC / ' ' /         ! File Carriage Control Character
      LOGICAL*1 DOLIST,ERR,AUTOMX
        DATA DOLIST / .TRUE. /
        DATA AUTOMX / .TRUE. /
      DATA PI / 3.141593 /
C
      KMAX = 200                ! Maximum Dimensioning of Arrays
C
C
      CALL VINIT
      CALL VSTR('EAA DATA DISPLAY PROGRAM',1,20,'B',1)
      CALL ISLEEP(0,0,2,0)
      CALL VPRINT(' by ')
      CALL VSTR('DRW',0,0,'BF',0)
      CALL VSTR('   Nov 1985',0,0,' ',2)
      CALL ISLEEP(0,0,2,0)
      CALL VPUT(4,1)
      CALL VPRINT('EAA Input Data File [.EA1] - ')
      CALL SCOPY('DATA.EA1',EFILE)
      CALL ASKNAM(EFILE)
      LDOT=INDEX(EFILE,'.')
      IF (LDOT.EQ.0) CALL CONCAT(EFILE,'.EA1',EFILE)
      CALL ASSIGN(2,EFILE)              ! Open EAA input on LUN 2
      CALL VPUT(7,1)
      LDOT=INDEX(EFILE,'.')
      CALL SCOPY(EFILE,OFILE,LDOT)
      CALL CONCAT(OFILE,'EN',OFILE)
      IF (EFILE(LDOT+3).EQ.'1') CALL CONCAT(OFILE,'1',OFILE)
      IF (EFILE(LDOT+3).EQ.'2') CALL CONCAT(OFILE,'2',OFILE)
      CALL VPRINT('EAA Output Data File - ')
      CALL ASKNAM(OFILE)
      CALL TRIM(OFILE)
C     LO=LEN(OFILE)
C     IF (LO.EQ.0) CALL SCOPY('NL:',OFILE)
      CALL ASSIGN(3,OFILE)
      CALL VPUT(10,1)
      CALL ASKYN('Do you want terminal listing?','Y',MAYBE)
      IF (MAYBE.EQ.'N') DOLIST=.FALSE.
      MODE=1
```

```
 16     CALL VPUT(11,1)
 17     K2=3
        CALL ASKI('First Useful EAA Channel : ',1,K2)
        IF (K2.LT.2 .OR. K2.GT.7) GOTO 17
        K1=K2-1                      ! Used for Totals, Averages
        CALL ISLEEP(0,0,1,0)
C
        CALL SETEAA
C
        CALL VCLEAR
        CALL V132C
C
11      FORMAT(A80)
C
        IF (.NOT.DOLIST) GOTO 99
        CALL VSCROL(3,24)
        CALL VPUT(1,1)
        CALL VBOLD
        TYPE 100, (I,I=2,10)
        TYPE 110, (EAADP(I),I=1,10)
        CALL VOFF
 99     WRITE(3,100) (I,I=2,10)
100     FORMAT(' EAA ',T10,I8,8I11,7X,'Total N')
        WRITE(3,110) (EAADP(I),I=1,10)
110     FORMAT(' t ',10F11.4)
C
111     KOUNT = 0
        DO 112 K=1,13
112       YMAX(K)=0.
C
C               READ IN HEADER LINE
C
        READ(2,*)
C
C               READ IN ENDING TIME AND CHANNEL VOLTAGES
C
115     READ(2,118,END=500,ERR=500) TIME,(V(K),K=1,10)
118     FORMAT(11F7.4)
        KOUNT = KOUNT + 1
        T(KOUNT)=TIME
125     CONTINUE
C
        CALL EAASIZ(V,PN,PV,TN,TV,TS,TD)
C
        DO 130 K=1,9
130      Y(KOUNT,K)=PN(K)
        DMOM0=0.                     ! Zeroth Moment of the Diameter Distribution
        DMOM1=0.
        DMOM3=0.
        DO 132 K=K1,9
          DMOM0=DMOM0+PN(K)
          DMOM1=DMOM1+PN(K)*EAADPM(K)
132       DMOM3=DMOM3+PN(K)*(EAADPM(K))**3
```

```
        Y(KOUNT,13)=DMOMO                    ! Total Number
        Y(KOUNT,12)=PI*DMOM3/6.              ! Total Volume
        Y(KOUNT,11)=-4.                      ! LOG(Volume mean diameter)
        Y(KOUNT,10)=-4.                      ! LOG(Number mean diameter)
        IF (DMOMO.EQ.0.) GOTO 133
        Y(KOUNT,10)=ALOG10(DMOM1/DMOMO) ! Number mean diameter (TD/TN)
        Y(KOUNT,11)=ALOG10(DMOM3/DMOMO)/3.        ! Volume mean diameter
C                      or (6/PI*TV/TN)**(0.33333333))
133     DO 135 K=1,13
135       IF (Y(KOUNT,K).GT.YMAX(K)) YMAX(K)=Y(KOUNT,K)
C
        IF (DOLIST) TYPE 140, TIME,(PN(I),I=1,9),TN
        WRITE(3,140) TIME,(PN(I),I=1,9),TN
140     FORMAT(1X,F8.4,9(1X,1P9E11.3))
C
        IF (KOUNT.GE.KMAX) GOTO 490          ! Arrays Full
        GOTO 115                             ! LOOP
C
C
490     CALL PRINT('Out of Array Space . . .')
500     CALL ISLEEP(0,0,2,0)
        TMIN=T(1)
        TMAX=T(KOUNT)
        CALL VSCROL(1,24)                    ! Normal Full Scrolling
        CALL PRINT(' ')
550     CALL VPUT(24,1)
        CALL QPRINT('EAA Channel (2-10 ; 1=tot ; 11=dp(Nav) ; ')
        CALL QPRINT('12=dp(Vav) ; 0=set max [N] ; 20=set times ; ')
        CALL QPRINT('13=Tot Vol ; -1=quit) ')
        CALL ASKI(':',1,IN)
        IF (IN.LT.0) GOTO 900
        IF (IN.GT.13) GOTO 700
        IF (IN.NE.0) GOTO 555
552     CALL VHLINE(24,'  ',1,100)
        CALL VPUT(24,1)
        YMAX2=YMAX1
        CALL ASKR('Enter Max [Number] to Plot [-1. permanent]:',1,YMAX2)
        IF (YMAX2.LT.0.) GOTO 800
        YMAX1=YMAX2
        GOTO 560
555     ICH=IN-1
        IF (IN.EQ.1) ICH=13                        ! Total Number
        IF (AUTOMX) YMAX1=YMAX(ICH)
560     CALL VINIT
        IF (ICH.NE.10.AND.ICH.NE.11) CALL VSETS(0.,YMAX1,TMIN,TMAX,1)
        IF (ICH.EQ.10.OR.ICH.EQ.11) CALL VSETS(-2.,0.,TMIN,TMAX,1)
        CALL VXPLOT(KOUNT,T,Y(1,ICH))
        CALL VPAUSE
        GOTO 550
700     CALL ASKR('Minimum Hours:',1,TMIN)
        CALL ASKR('Maximum Hours:',1,TMAX)
        DO 720 K=1,13             ! Reset Maximums for new time range
720     YMAX(K)=-4.
```

```
      DO 750 J=1,KOUNT
      IF (T(J).LT.TMIN) GOTO 750
      IF (T(J).GT.TMAX) GOTO 760
      DO 735 K=1,13
735     IF (Y(J,K).GT.YMAX(K)) YMAX(K)=Y(J,K)
750   CONTINUE
760   CONTINUE
      GOTO 560
800   AUTOMX=.NOT.AUTOMX
      GOTO 560
900   CALL V80C
      STOP 'VTEAA Program Finished'
      END
```

```
        PROGRAM SAVOPC            ! June 85 by DRW

C       This program builds an ASCII file of OPC counts.
C       Designed for convenient user input.

C       LINK with RTLIB

        DIMENSION X(50),Y(50),KOUNT(16),ITIM(4),COUNT(16)
        BYTE FNAME(20),RUNID(40),INPUT(12),IN
        EQUIVALENCE(INPUT(1),IN)
        BYTE YES,OK,MAYBE
        REAL*8 SST3,SST
        LOGICAL*1 ERR,SKIP,ASKT
        DATA YES /'Y'/ , OK /' '/
        DATA NSEC / 120 /
C
        STIME(KHR,KMI,KSE)= KSE+60.*(KMI+60.*KHR)
C
 5      CALL SCOPY('NEWOPC.OPC',FNAME)
        CALL ASKNAM(FNAME)
        CALL ASSIGN(3,FNAME,0,'NEW')
        CALL QPRINT('RUN NAME : ')
        ACCEPT 84, RNAME
        CALL QPRINT('DATE (Mo,Da,Yr) : ')
        ACCEPT *, IMON,IDAY,IYEAR
        CALL QPRINT('Enter a Half-Line of Run I.D.: ')
        ACCEPT 80, RUNID
        WRITE(3,88) FNAME,RNAME,IMON,IDAY,IYEAR,RUNID
C
        CALL QPRINT('Initial Printout Time (Hr:Mi:Se) : ')
        ACCEPT 16, KHR,KMI,KSE
16      FORMAT(I2,':',I2,':',I2)
        CTIME=STIME(KHR,KMI,KSE)
        CALL ASKI('Interval (INTEGER Seconds) :',1,NSEC)

        CALL VINIT
        CALL VSCROL(7,24)
        CALL VPUT(1,1)
        CALL VPRINT('Please Enter OPC Counts (as INTEGER or REAL)')
        CALL VPUT(3,1)
        CALL VPRINT('Use the following Codes instead of Counts:')
        CALL VSTR('T',5,1,'B ',1)
        CALL VSTR(' = Change Time   ',0,0,' ',1)
        CALL VSTR('-',0,0,'B ',1)                        ! - or R
        CALL VSTR(' = Reenter Data   ',0,0,' ',1)
        CALL VSTR(',',0,0,'B ',1)                        ! , or Z
        CALL VSTR(' = Zero Rest of Set',0,0,' ',1)
        CALL VSTR('   E',0,0,'B ',1)
        CALL VSTR(' = Exit',0,0,' ',0)
        CALL VPUT(7,1)

        SKIP=.FALSE.
        NSET=1
        CALL NUTIME(KHR,KMI,KSE,CTIME,-NSEC)     ! True Starting Time
20      OTIME=CTIME
        CALL NUTIME(KHR,KMI,KSE,CTIME,NSEC)
22      SST=SST3(KHR,KMI,KSE)
        ASKT=.FALSE.

24      TYPE 25, SST,NSEC
25      FORMAT('$',A8,5X,I5,' Sec ;  OK ? ')
        CALL GETSTR(5,INPUT,12,ERR)
        IF (ERR) STOP 'ERROR ON GETSTR'

        MAYBE=IN
        IF (MAYBE.EQ.OK.OR.MAYBE.EQ.'Y'.OR.MAYBE.EQ.0) GOTO 100 ! TIME OK
        IF (MAYBE.EQ.'N'.OR.MAYBE.EQ.'T') GOTO 26
        SKIP=.TRUE.
        GOTO 100

26      IF (ASKT) GOTO 30                   ! Ask for Interval or Time?
        CALL QPRINT('Interval (INTEGER Seconds) :')
        CALL GETSTR(5,INPUT,12,ERR)
        CALL VERIFY(INPUT,'0123456789.-',K)      ! Legitimate #?
        IF (K.EQ.0) GOTO 27
        IF (IN.EQ.'E') GOTO 500
        GOTO 26                                  ! Repeat Request
27      L=LEN(INPUT)
        DECODE (L,28,INPUT) NSEC
28      FORMAT(I)
        IF (NSEC.LT.0) GOTO 500
        CTIME=OTIME
        CALL NUTIME(KHR,KMI,KSE,CTIME,NSEC)
        SST=SST3(KHR,KMI,KSE)
        ASKT=.TRUE.
        GOTO 24

30      TYPE 35
35      FORMAT('$TIME (HR:MI:SE) : ')
        ACCEPT 16, KHR,KMI,KSE
        CTIME=STIME(KHR,KMI,KSE)
        GOTO 22          ! Make Sure Times Are Right


90      TYPE 95, SST
95      FORMAT(' Re-entry for Time ',A8)

C                 Main OPC Data Input Loop

100     DO 200 I=1,16
         IF (SKIP) GOTO 120
110      TYPE 115, I
115      FORMAT('$Ch',I3,' : ')
         CALL GETSTR(5,INPUT,12,ERR)
120      SKIP=.FALSE.
```

```
        CALL VERIFY(INPUT,'0123456789.',K)      ! Legitimate #?
        IF (K.EQ.0) GOTO 140                    ! Legal # Input
        IF (IN.EQ.'-'.OR.IN.EQ.'R') GOTO 90     ! ReEnter
        IF (IN.EQ.','.OR.IN.EQ.'Z') GOTO 300    ! Zero for Rest
        IF (IN.EQ.'T'.OR.IN.EQ.'C') GOTO 26     ! Time Wrong
        IF (IN.EQ.'E') GOTO 500                 ! Exit
        GOTO 110                                ! Repeat Input
140     CALL VERIFY(INPUT,'0123456789-',K)      ! Decimal there?
        IF (K.EQ.0) CALL CONCAT(INPUT,'.',INPUT,12)     ! Add it
        L=LEN(INPUT)
        DECODE (L,145,INPUT) COUNT(I)
145     FORMAT(F)
        IF (COUNT(I).LT.0.) GOTO 90     ! User acknowledges mistake
200     CONTINUE

55      FORMAT(I9)
56      FORMAT(F9.0)

C               WRITE OUT OPC RECORD TO DATA FILE

220     SST=SST3(KHR,KMI,KSE)
        WRITE(3,250) SST,NSEC
250     FORMAT(A8,5X,I5)
        WRITE(3,255) (COUNT(I),I=1,8)
        WRITE(3,255) (COUNT(I),I=9,16)
255     FORMAT(8F9.0)
        NSET=NSET+1
        GOTO 20

300     DO 310 J=I,16
          COUNT(J)=0.
310     CONTINUE
        GOTO 220

500     CALL CLOSE(3)
        CALL FPRINT(' ')
        CALL ASKYN('Create Another Output File?','Y',MAYBE)
        IF (MAYBE.EQ.'Y') GOTO 5
        CALL VINIT
        STOP 'SAVOPC done.'

80      FORMAT(80A1)
81      FORMAT(A1)
84      FORMAT(A4)
88      FORMAT(20A1,2X,A4,2X,I2,'/',I2,'/',I2,2X,40A1)
        END
        PROGRAM SAVTOL            ! July 85 by DRW

C       This program builds a ZPLOTTABLE ASCII file of Toluene data.
C       Designed for convenient user input.

C       LINK with RTLIB

        BYTE FNAME(20),DRIFT
        REAL SLOP(2),YINT(2),TIMT(2),TOLU(2)
        REAL*8 SSTH,SST

C       DTIME = Time of Day in REAL Hours
C       CYCLE = Cycle Time in REAL Hours
C       SST = Time of Day in 'hh:mm:ss' (not genuine string)

        STIME(KHR,KMI,KSE)= KHR+KMI/60.+KSE/3600.

10      CALL SCOPY('NEWTOL.TOL',FNAME)
        CALL ASKNAM(FNAME)
        CALL ASSIGN(3,FNAME,0,'NEW')

        DO 150 I=1,2
C       CALL ASKR('Enter SLOPE (ppm/chart div)','N',SLOPE)
C       CALL ASKR('Enter YINTER (ppm tol)','N',YINTER)
        TYPE 12
12      FORMAT('$Enter SLOPE (ppm/chart div): ')
        ACCEPT *, SLOP(I)
        TYPE 14
14      FORMAT('$Enter YINTER (ppm tol): ')
        ACCEPT *, YINT(I)
C
        IF (I.NE.1) GOTO 140
        CALL ASKYN(' Time Interpolation for GC drift?','N',DRIFT)
        IF (DRIFT.EQ.'N') GOTO 160
140     TYPE 145, I
145     FORMAT('$Enter CAL TIME #',I1,' (hh:mm:ss): ')
        ACCEPT 16, KHR,KMI,KSE
        TIMT(I)=STIME(KHR,KMI,KSE)
150     CALL FPRINT(' ')
        DT=TIMT(2)-TIMT(1)             ! Delta Time
160     NSET=1

        CMIN=2.6                       ! Toluene GC cycle time
        CALL ASKR('Interval (REAL Minutes) :',1,CMIN)
        IF (CMIN.NE.0.) GOTO 13
        CMIN=2.6
        TYPE *, 'ASKR Default not working.'
13      CYCLE=CMIN/60.

15      CALL QPRINT('New Inject Time (Hr:Mi:Se) : ')
        ACCEPT 16, KHR,KMI,KSE
16      FORMAT(I2,':',I2,':',I2)
```

```
      DTIME=STIME(KHR,KMI,KSE)

      CALL VINIT
      CALL VSCROL(7,24)
      CALL VPUT(1,1)
      CALL PRINT('  Please Enter Toluene Peak Heights (as REAL)')
      CALL VPUT(3,1)
      CALL VPRINT('The following Codes may be used in place ')
      CALL PRINT('of Heights:')
      CALL VSTR('-1.',5,1,'B ',1)
      CALL VSTR(' = Change Time   ',0,0,' ',1)
      CALL VSTR('0.',0,0,'B ',1)
      CALL VSTR(' = Skip   ',0,0,' ',1)
      CALL VSTR('-2.',0,0,'B ',1)
      CALL VSTR(' = Multiskip   ',0,0,' ',1)
      CALL VSTR('-9.',0,0,'B ',1)
      CALL VSTR(' = Exit',0,0,' ',0)
      CALL VPUT(24,1)
      GOTO 22                         ! No Need to Cycle from New

20    DTIME=DTIME+CYCLE
22    SST=SSTH(DTIME)
24    TYPE 25, DTIME,SST
25    FORMAT('$',F8.4,' (',A8,')    Height: ')
      CALL VCSAV
      ACCEPT *, HT
      IF (HT.LE.0.) GOTO 300

C               Main Toluene Data Input

100   CALL VCRES
      CALL VUP(1)
      TOLU(1) = SLOP(1) * HT + YINT(1)
      TOLU(2) = SLOP(2) * HT + YINT(2)
      TOL=TOLU(1)
      IF (DRIFT.EQ.'Y') TOL=TOL+(DTIME-TIMT(1))*(TOLU(2)-TOLU(1))/DT
      TYPE 105, HT,TOL
105   FORMAT(1X,F5.2,4X,F7.3)

C               WRITE Toluene Data to ASCII File

220   IF (DRIFT.EQ.'N') WRITE(3,250) DTIME,TOL,HT
      IF (DRIFT.EQ.'Y') WRITE(3,260) DTIME,TOL,HT,TOLU(1),TOLU(2)
250   FORMAT(1X,F8.4,2X,F8.4,2X,F6.2)
260   FORMAT(1X,F8.4,2X,F8.4,2X,F6.2,2X,F8.4,2X,F8.4)
      NSET=NSET+1
      GOTO 20

C     Change Times Here

300   IF (HT.EQ.0.) GOTO 20                        ! Skip One Cycle
      IF (HT.EQ.-9.) GOTO 500                      ! Exit
      IF (HT.EQ.-2.) GOTO 350
      GOTO 15

350   NSKIP=1
      CALL ASKI('Number of Cycles to Skip',1,NSKIP)
      DTIME=DTIME+NSKIP*CYCLE
      GOTO 22

500   CALL CLOSE(3)
      CALL FPRINT(' ')
      CALL ASKYN('Create Another Output File?','N',MAYBE)
      IF (MAYBE.EQ.'Y') GOTO 10
      CALL VINIT
      STOP 'SAVTOL done.'

      END
```

*** RTLIB ***

RTLIB consists of over 120 routines written for the PDP-11/03 minicomputers running with an RT-11 operating system. The RTLIB subroutines deal with video terminal display (VT100 family), data plotting, data sampling, time conversion, timed sampling, EAA control, and statistics. The subroutines are available in a master object library RTLIB.OBJ on the RTLIB system disk. Source code (predominantly FORTRAN IV with a few MACRO assembly language routines) is distributed among about two dozen files on the RTLIB Source disk. Check RTLIB.COM and the files it references to see how and from what RTLIB is generated. RTLIB is a composite of the object libraries VXLIB, ADLIB, TLIB, SLIB, WLIB, and EAALIB. See the documentation (.DOC) files under those names for further (older) information on the available subroutines; check the command files (.COM) to obtain the names of the source code files which were used to create the object libraries.

Also see specific sampling and plotting programs (such as WATCH, ADPLOT, DOEAA, DO2EAA, and DATEST) which link with RTLIB for examples of usage.

Dale Warren (DYW)
Caltech / ChE x4671
Novermber 15, 1985

** RTLIB Subroutines by Function **

==> VIDEO PLOTTING:

	RANGE	Simply finds the minimum and maximum values of an array.
	RANGES	Finds the minimum and maximum values of the X and Y arrays.
	VADD	Adds points to the video screen plot.
	VBORD	Generates the Border of a plot (directly to the VT).
	VSETE	Sets extent of screen to be used for plotting.
*	VSETS	Sets range of plot in user units.
	VSETT	Sets number and spacing of ticks and border.
	VSHIFT	Moves SCREEN plot view one division (tick) to the right.
	VSINIT	Initializes the SCREEN matrix for plotting.
	VTICKS	Displays ticks and labels on the video screen plot.
	VTITLE	Displays the pre-selected title onto the VT100.
*	VXADD	Adds data points to a VT100 plot (VXPLOT).
*	VXPLOT	Generates a plot of X-Y data onto the video screen.
	VXPLT1	Generates a plot of equispaced data onto the VT100.
	VXSHOW	Redraws active window on video plot from SCREEN matrix.

==> LOW LEVEL VT100 COMMANDS:

	DSPLY	Puts a single character onto VT100 at selected position.
	V132C	Sets VT100 family terminal to 132 characters screen width.
	V80C	Sets VT100 family terminal to 80 characters screen width.
	VAT	Sets video attributes for string.
	VBLINK	Causes future VT output characters to blink (Flash).
	VBOLD	Causes future VT output characters to be in Boldface (bright).
	VCHSET	Establishes standard character sets on the VT100 terminal.
	VCLEAR	Simply clears the video screen.
	VCRES	Restores Cursor Attributes such as position (from last VCSAV).
	VCSAV	Saves Cursor Attributes such as position (for VCRES later).
	VDOWN	Moves video cursor down a selected number of positions.
	VHLINE	Draws a horizontal line onto the VT100, using graphics.
	VHOME	Simply moves the video cursor home (top left of screen).
*	VINIT	Initializes the video terminal, with following CALLs:
	VLED	Controls terminal LEDs (1 on VT100s, 4 on ADM36 keyboard).
	VLEFT	Moves video cursor left a selected number of positions.
	VOFF	Causes future VT output characters to have normal attributes.
	VPUT	Move Video Cursor to selected position.
	VPUTST	Puts a string onto VT100 at selected location.
	VRESET	Resets video terminal to power-up state.
	VRIGHT	Moves video cursor right a selected number of positions.
	VREV	Causes future VT output characters to be reverse video mode.
	VSCROL	Sets scolling region of video screen (at least two lines).
	VSETG0	Use Standard Character Set G0 (Normally US ASCII) on VT100.
	VSETG1	Use Alternate Character Set G1 (Normally Graphics) on VT100.
*	VSTR	Advanced Video String Placement Routine.
	VUNDER	Causes future VT output characters to be underlined.
	VUP	Moves video cursor up a selected number of positions.
	VVLINE	Draws vertical line onto video screen, using graphics.

==> TIMED SAMPLING:

	ADSET		Quickly takes a set of A/D readings on one channel. (MACRO)
	CLOCK		Wait specified time while displaying time & accepting input.
	icmkt	+	Cancels an unexpired timed completion request.
	itimer	+	Schedules FORTRAN completion routine after specified interval.
	isched	+	Schedules FORTRAN completion routine at specified time of day.
	isleep	+	Program sleeps for specified (integer foursome) time interval.
	itwait	+	Program waits for specified internal time interval.
	iuntil	+	Program waits until specified (integer foursome) time of day.
	MDELAY		Delay Loop to waste MILSEC milliseconds after ADREAD. (MACRO)
	mrkt	+	Schedule MACRO completion routine after elapsed internal time.
	MSWAIT		Delay Loop to waste specified number of milliseconds. (MACRO)
	SDELAY		Waits for a specified number of seconds.
*	VREADM		Reads selected A/D Channel repeatedly with millisecond waits.
	VREADS		Reads selected A/D Channel repeatedly and quickly.
*	VREADW		Reads selected A/D Channel at system clock intervals.
	VREADX		Repeatedly reads A/D channel and then does statistics.
*	VSCAN		Flexible multichannel A/D Sampler with Statistics.

==> EAA SAMPLING:

CHEAAT	Allows user to CHange EAA Times for waiting.

```
   EAACOM       Processes character commands entered during EAA sampling.
   EAASIZ       Returns sectional aerosol size distribution from EAA dataset.
*  RESET        Grounds control signal DS0 momentarily.  RESETs EAA.   (MACRO)
   SETEAA       Sets up COMMON for simple EAA data interpretation.
*  STEP         Grounds DS3 control signal momentarily.  STEPs EAA.    (MACRO)

==>     SIMPLE A/D SAMPLING:

*  ADREAD       Read Selected A/D Channel Once.                        (MACRO)
   ASKADC       User-friendiy query for the range of the A/D.
   PGA          Reads A/D using selected PGA range.
   VAD          Reads A/D voltage once, using autoranging of PGA.
*  VREAD        Read selected A/D Channel once, returning voltage.

==>     D/A SIGNALS:

   DAC0         Sends raw value to Digital-to-Analog Converter #0 to output.
   DAC1         Sends raw value to Digital-to-Analog Converter #1 to output.
*  VDAC0        Sets Digitial-to-Analog Converter #0 to selected voltage.
*  VDAC1        Sets Digitial-to-Analog Converter #1 to selected voltage.

==>     TIME CONVERSION:

   CVHRST       Converts hours to a full time string.
   CVHRT4       Converts hours to a integer time foursome.
   CVT4ST       Converts integer time foursome to full time string.
   cvttim     + Converts internal time to an integer time foursome.
* =ELAPSE       Calculates elapsed seconds between two internal times.  R4   >
*  gtim       + Gets time of day as internal time.
  =HRSST        Converts short ASCII time into real hours.              R4   >
  =HRSTR        Converts time string into real hours.                   R4   >
  =HRT4         Converts time foursome into real hours.                 R4   >
  =HRT4X        Converts expanded time foursome into real hours.        R4   >
* =HRTINT       Converts Internal Time to Real Hours.                   R4   >
   jtime      + Converts integer time foursome into internal time.
   NEWTIM       Puts current time into multiple forms stored in COMMONs.
   NOWIS        Types out current time of day: " The time is now hh:mm:ss"
   NUTIME       Update real seconds by integer seconds, into time threesome.
  =secnds     + Returns current time past midnight in seconds.          R4   >
  =SECT4        Converts integer time foursome into real seconds.       R4   >
  =SST3         Converts time threesome to short ASCII time hh:mm:ss.   A8   >
  =SSTH         Converts real hours to short ASCII time, hh:mm:ss.      A8   >
  =SSTS         Converts real seconds to short ASCII time, hh:mm:ss.    A8   >
   T4ADD        Adds two time foursomes together.
   TCHECK       Checks if internal time is within given hours range.
   TIME4        Makes integer time foursome from its expanded form.
   TIME4X       Converts integer time foursome to its expanded form.
*  timasc     + Converts an internal time into short ASCII time.
   time       + Returns current time as a short ASCII time.
   TIMEIS       Types out " Elapsed time is hh:mm:ss at Time hh:mm:ss."
   TIMEX        Converts Internal Time into multiple forms stored in COMMONs.
  =TINTHR       Converts real hours into Internal Time.                 J4   >
  =TINTST       Converts full time string into Internal Time.           J4   >
```

!	WHEN		Internal Time minus Real Ticks into Elapsed Time String.

==> STRINGS & TERMINAL INPUT/OUTPUT:

	ASKFIL		Asks for a filename and opens the file.
*	ASKI		Asks for an integer value, with prompt and default.
*	ASKNAM		Asks for a filename from the user.
*	ASKQ		Asks a question, allowing defaults, accepting 1 char answer.
*	ASKR		Asks for a real value, with prompt and default.
*	ASKSTR		Asks for a string, with prompt, default, length, and spacing.
	ASKTIM		Accepts a time foursome as hh:mm:ss:tt from the user.
*	ASKYN		Asks a Y/N question, allowing defaults, and accepts answer.
	concat	+	Concatenates two variable length strings.
	FPRINT		Prints out string using normal Fortran Formatting, with CRLF.
	getstr	+	Reads a character string from specified logical unit.
	gtlin	+	Transfers a line of input to the program.
	index	+	Returns the location of a substring in a string.
	insert	+	Replaces a portion of one string with another string.
	=INTYPE		Determines what a string represents, if is valid input. I2 >
	=ipeek	+	Returns value of word at specified absolute address.
	ipoke	+	Stores an integer value at an absolute memory location.
	=iscomp	+	Compares two character strings.
	=ittinr	+	Gets one character from the console terminal.
	=ittour	+	Transfers one character to the console terminal.
	=iverif	+	Indicates whether characters in one string appear in another.
*	KEMODE		Controls mode of terminal input, will disable waiting.
*	KEYCHK		Accepts Keyboard Input, one character at a time.
	KEYIN		Reads in a single character. (LF of CRLF is omitted.)
*	=len	+	Returns the number of characters in a specified string. I2 >
*	LOCASE		Disables automatic conversion of terminal input to upper case.
	LPRINT		Prints an ASCII string at the lineprinter (with CRLF).
	LPRNT1		Prints an ASCII string at the lineprinter (without CRLF).
*	print	+	Outputs an ASCII string to the terminal.
	putstr	+	Writes a character string onto a specified logical unit.
	QPRINT		Prints out string using Fortran Formatting, without CRLF.
	repeat	+	Repeatedly self-concatenates a string to specified length.
	scomp	+	Compares two character strings.
*	scopy	+	Copies a character string from one array to another.
	SETJSW		Sets or clears a selected bit in the Job Status Word.
!	SHOJSW		Types out Job Status Word pattern in Octal.
	SPECIN		Waits for a single character in special input mode.
	strpad	+	Pads a variable length character string with ending blanks.
	substr	+	Copies a substring from a specified string.
	transl	+	Replaces a string with another after substituting characters.
	TREADY		Clears IOFLAG in /TINPUT/ so new terminal input can be used.
	trim	+	Removes trailing blanks from a character string.
	UPCASE		Forces automatic conversion of terminal input to upper case.
	verify	+	Indicates whether characters in one string appear in another.
*	VPRINT		Prints a character string, suppressing the carriage return.
	VPAUSE		Waits at bottom of plot for user to enter command character.

==> STATISTICS:

```
*  DOSTAT      Calculates simple statistics for M-th set of summations.
   ISORT       Orders an integer array, lowest to highest.
   ISTAT       Calculates simple statistics for an integer array of data.
   STAT        Calculates simple statistics for a real array.
*  STATAD      Add point to Mth summation set for statistics.
   STATS       Calculates simple statistics given three summations.
   STINIT      Initialize M-th summation set for statistics.
```

Note that entries that are in lower case and denoted by a + are in SYSLIB, and documentation may be found in the Programmers' Reference Manual. Entries denoted by a * are most likely to be useful.

```
      **      RTLIB Subroutines by Alphabetical Listing        **

ADREAD        Read Selected A/D Channel Once.                         (MACRO)
      ICHAN   A/D Channel Number [0-15 normally]                        I2  <=
      ISIG    Raw A/D Output Number [0-4095]                            I2   >
        !     Voltage = V_fullscale/4096. * ISIG
        !     The full scale voltage depends on the A/D board.
        !     To use PGA, add offset to ICHAN of 64. for 2X,
        !       128. for 5X, 192 for 10X.

ADSET         Quickly takes a set of A/D readings on one channel.     (MACRO)
      ICHAN   A/D Channel Number [0-15 normally]                        I2  <=
      IARRAY  Array for Raw A/D Readings (at least KOUNT in size)       I*   >
      KOUNT   Number of A/D Readings to take                            I2  <=
        !     This routine is fast -- 18K per second.

ASKADC        User-friendly query for the range of the A/D.
        /ADCALO/ VCON
```

```
         +        > ?
          !       This routine sets VCON for any voltage reading routines.
          !       Needed since our two RT-11 A/D's have different full scales.
          !       Defaults to R (5V) or K (10V) depending on version of RTLIB.

ASKFIL            Asks for a filename and opens the file.
       IO         Logical Channel Number for the file.                  I2  <=
       FILE       Filename String.  May be preset to default for CR.    B20 <>
        +         > ASKNAM, assign

ASKI              Asks for an integer value, with prompt and default.
       PROMPT     Prompt String, holding the question.                  B*  <=
       DEF        0 if no default, 1 if IVAL holds default, -1 for zero. B1  <=
       IVAL       Integer Value to be input.  May have preset default.  I2  <>
        +

ASKNAM            Asks for a filename from the user.
       FILE       Filename String.  May be preset to default for CR.    B20 <>
        +         > scopy, trim

ASKQ              Asks a question, allowing defaults, accepting 1 char answer.
       PROMPT     Prompt String, holding the question.                  B*  <=
       DEF        Default answer (anything; if 0 there is no default)   B1  <=
       ASK        The user's answer; must be a single character.        B1   >
        +         > QPRINT
         !        If user hits RETURN only, ASK=DEF unless DEF=0,
         !         in which case the prompt is typed again.

ASKR              Asks for a real value, with prompt and default.
       PROMPT     Prompt String, holding the question.                  B*  <=
       DEF        0 if no default, 1 if VALUE is default, -1 for zero.  B1  <=
       VALUE      Real value to be input.  May be preset to default.    R4  <>
        +

ASKSTR            Asks for a string, with prompt, default, length, and spacing.
       PROMPT     Prompt String, holding the question.                  B*  <=
       DEF        0 if no default, 1 if STRING is default, -1 for zero. B1  <=
       STRING     String to be input.  May be preset default.           B*  <>
       IPOS       Position at which ':' after default should be placed. I2  <=
       MAXLEN     Maximum number of characters to be input into STRING. I2  <=
        +
       !          Note that IPOS=0 uses natural spacing for the input line.
       !          If MAXLEN=0, the maximum length of STRING defaults to 20.

ASKTIM            Accepts a time foursome as hh:mm:ss:tt from the user.
       ITIM       Integer time foursome (IH,IM,IS,IT).                  I8   >
       ICHECK     Set nonzero to ask for confirmation of time input.    I2  <=
        +         > ASKYN, CVT4ST, QPRINT

ASKYN             Asks a Y/N question, allowing defaults, and accepts answer.
       STRING     Prompt String, holding the question.                  B*  <=
       DEF        Default answer (Y or N or 0 if none; anything allowed) B1  <=
       ASK        The user's answer.  Must be Y or N or DEF.            B1   >
```

```
      +       > ASKQ

CHEAAT        Allows user to CHange EAA Times for waiting.
     TIMES    Array of ten delay times in seconds, channels 2-10.    R4  <>
      +       > ASKYN

CLOCK         Wait specified time while displaying time & accepting input.
     SEC      Seconds to wait in this subroutine.                     R4  <=
     LDAY     Flag to show time of day in upper left:                 I2  <=
      .        0 off, 1 on, 2 flash, -1 set elapsed time zero
     LRUN     Flag to show elapsed time from some reference:          I2  <=
      .        0 off, 1 on, 2 flash, negative for remaining (-) time.
     LWAIT    Flag to show time of this CLOCK wait:                   I2  <=
      .        0 off, 1 on, 2 flash, negative for remaining (-) time.
     KEYON    Flag telling whether keyboard input will be accepted:   I2  <=
      .         0 No input allowed
      .         1 Char input calls KEYMON , -1 Char input just saved
      .         2 Line input calls KEYMON , -2 Line input just saved.
      .         3 Char input where any keystroke causes CLOCK to RETURN.
     KEYMON   Subroutine name that may be called for positive KEYMON. .   <=
      /TINPUT/ IOFLAG,IOMODE,TINPUT()
      /CLOCK0/
      /CLOCK1/
      +       > ELAPSE, KEMODE, KEYCHK, SSTS, VCRES, VCSAV, VPRINT, VPUT
      +       > VSTR, gtim, timasc
       !      This is a complicated subroutine for civilized waiting.

CVHRST        Converts hours to a full time string.
     HRS      Real hours.                                             R4  <=
     TSTR     Full ASCII time string, hh:mm:ss:tt, 0 byte terminator. B12  >
      +       > CVHRT4, CVT4ST

CVHRT4        Converts hours to a integer time foursome.
     HRS      Real hours.                                             R4  <=
     ITIM     Integer time foursome (IH,IM,IS,IT)                     I8   >
      +       >  TINTHR, cvttim

CVT4ST        Converts integer time foursome to full time string.
     ITIM     Integer time foursome (IH,IM,IS,IT)                     I8  <=
     TSTR     Full Time string, 'hh:mm:ss:tt', terminated by 0 byte.  B12  >
      +       > transl

DAC0          Sends raw value to D-to-A Converter #0 to output.      (MACRO)
     IVAL     Raw value for DAC0 to output [0-4095]                   I2  <=
       !      Current IVAL range 0-4096 equals -10 to +10 V.

DAC1          Sends raw value to D-to-A Converter #1 to output.      (MACRO)
     IVAL     Raw value for DAC1 to output [0-4095]                   I2  <=
       !      Current IVAL range 0-4096 equals -10 to +10 V.

DOSTAT        Calculates simple statistics for M-th set of summations.
     M        Data Set Number, currently 1 to 40.                     I2  <=
     NX       Number of data values that were in summation set M.     I2  <=
        AVER    Average of values in data set M.                  R4    >
        STDEV   Standard deviation of values in data set M.       R4    >
        SLOPE   Slope, by linear regression (per point) for data set M. R4    >
         /STAT/ N(40),S1(40),S2(40),SP(40)
         +      STATS
          !     Used with STINIT and STATAD to handle summations.

 DSPLY          Puts a single character onto VT100 at selected position.
        LINE    Line number (1-24).                                I2   <=
        ICOL    Column number (1-132).                             I2   <=
        CHAR    Character.                                         B1   <=
         +      > VPUT

 EAACOM         Processes character commands entered during EAA sampling.
        CHAR    Character that was input by user.                  B1   <=
         /KILL/ KILL
         +      > SPECIN, VHLINE, VHOME, VPRINT, VPUT, VSCROL, ittour
          !     Recognized options are:
          !      K for immediate Kill of EAA acquisition (program STOP)
          !      L to make this the Last cycle (sets KILL to 1)
          !      C to Continue sampling (sets KILL to -1)
          !      H to print brief Help listing these options.
          !     User program must watch the /KILL/ COMMON after each
          !      EAA cycle to make this work.

 EAASIZ         Returns sectional aerosol size distribution from EAA dataset.
        V       Array of EAA voltages (picoamps), channels 2-11.   R40 <=
        PN      Array of EAA number concentrations, channels 2-10. R36  >
        PV      Array of EAA volume concentrations, channels 2-10. R36  >
        TN      Total number concentration (#/cc).                 R4   >
        TV      Total volume concentration (cubic microns per cc). R4   >
        TS      Total surface area (square microns per cc).        R4   >
        TD      Number-diameter product (microns per cc).          R4   >
         /EAA/ EAADP(10),EAADPM(9),EAACON(9),EAASN(9),EAAVN(9)
          !     This is the trivial inversion approach neglecting
          !      any EAA channel cross-sensitivity.  See SETEAA,
          !      which must be called first to set up /EAA/.
          !     Note PV array has units of cubic microns per cc, which
          !      equals micrograms per cubic meter at unit density.

=ELAPSE         Calculates elapsed seconds between two internal times.  R4   >
        START   Initial Internal Time.                             J4   <=
        FINISH  Ending Internal Time.                              J4   <=
         +      > ajflt, jjcvt, jsub

 FPRINT         Prints out string using normal Fortran Formatting, with CRLF.
        STRING  Byte array to be printed out (terminated by 0 byte).   B*   <=
          ! TYPEs out STRING(I),I=1,LEN(STRING) to FORMAT(' ',80A1)

=HRSST          Converts short ASCII time into real hours.         R4   >
        SST     Short ASCII time, hh:mm:ss.                        A8   <=
         +      > HRT4X
```

```
=HRSTR          Converts time string into real hours.                 R4   >
       TSTR     Time string, 'hh:mm:ss:tt', terminated with 0 byte.   B12 <=
        +       > HRT4X

=HRT4           Converts time foursome into real hours.               R4   >
       ITIM     Integer time foursome (IH,IM,IS,IT)                   I8  <=
        +       > HRT4X

=HRT4X          Converts expanded time foursome into real hours.      R4   >
       IH       Integer hours.                                        I2  <=
       IM       Integer minutes.                                      I2  <=
       IS       Integer seconds.                                      I2  <=
       IT       Integer ticks.                                        I2  <=

=HRTINT         Converts Internal Time to Real Hours.                 R4   >
       TINT     Internal Time.                                        J4  <=
        +       > jmov, jjcvt, ajflt

=INTYPE         Determines what a string represents, if is valid input. I2   >
       STRING   Character String.  Must terminate with zero byte.     B*  <=
        +       > iverif
         ! intype tells what string probably represents:
         !       -1 empty, 0 ordinary character string, 1 integer,
         !        2 F-format real  , 3 E-format real  , 4 G-format real
         !       INTYPE is not foolproof, future DECODE may still fail.
         !       INTYPE is normally used to check for valid input.

 ISORT          Orders an integer array, lowest to highest.
       N        Number of values in the array.                        I2  <=
       IRAY     Initial Integer array.                                I*  <=
       NRAY     Ordered Integer array.                                I*   >

 ISTAT          Calculates simple statistics for an integer array of data.
       N        Number of integer values (data points).               I2  <=
       IVAL     Integer array of values.                              I*  <=
       AV       Average of values.                                    R4   >
       SD       Standard deviation of data values.                    R4   >
       SL       Slope of values, by linear regression (per point).    R4   >
        +       > STATS

 KEMODE         Controls mode of terminal input, will disable waiting.
       MODE     Sets which of the following modes will be in effect:  I2  <=
        .        =0 restores normal mode, waits for each line entered
        .        =1 single character input, no echo, no waiting
        .        =2 line input mode, with echo, no waiting
        .        =-1 single character input, no echo, waits for character
        +       > ipeek, ipoke
         !       MODE=0 or -1 will wait for user to enter input.
         !       MODE=1 or 2 allow optional input, as user must be
         !       ready to handle no character available conditions.

 KEYCHK         Accepts Keyboard Input, one character at a time.
       CHAR     Latest available character input at the terminal.     B1   >
```

```
         /TINPUT/ IOFLAG,IOMODE,TINPUT(81)
          +       > ittinr, print
           !       This routine is intended to be the standard special
           !        keyboard input routine.  It will accumulate input
           !        by characters or lines, using and setting flags
           !        in /TINPUT/ to handle terminal input.  If IOFLAG
           !        is positive, there is terminal input available.
           !        KEYCHK need not wait for input, may return with
           !        CHAR=-1 if no character available.
           !        See also KEMODE and TREADY.

KEYIN             Reads in a single character.  (LF of CRLF is omitted.)
        CHAR      Latest available character input, if any.           B1    >
          +       > ittinr
           !       Intended to be used after KEMODE(1) so terminal input
           !       is immediate, with no waiting and no echo.
           !       If CHAR<0 there is no character available.

LOCASE            Disables automatic conversion of terminal input to upper case.
          +       > SETJSW
           !       This routine is used to allow program to accept lower
           !        case input.  Normally RT-11 automatically converts terminal
           !        terminal to upper case.  Use UPCASE to revert to normal.

LPRINT            Prints an ASCII string at the lineprinter (with CRLF).
        STRING    Character string to be printed.                     B*   <=

LPRNT1            Prints an ASCII string at the lineprinter (without CRLF).
        STRING    Character string to be printed.                     B*   <=

MDELAY            Delay Loop to waste MILSEC milliseconds after ADREAD. (MACRO)
        MILSEC    Number of milliseconds to wait.                     I2   <=
          !        MDELAY is MSWAIT shortened to give proper timing if
          !         ADREADs are timed by MDELAYs, by compensating
          !         for the temporal overhead in an ADREAD.

MSWAIT            Delay Loop to waste specified number of milliseconds. (MACRO)
        MILSEC    Number of milliseconds to wait.                     I2   <=
          !        Empirical PDP11/03 timing to within 1%.  This
          !         routine needs F/B monitor, and will not be accurate
          !         if completion routines or Foreground take over.

NEWTIM            Puts current time into multiple forms stored in COMMONs.
          +       > TIMEX , gtim
          !       See TIMEX for more information.

NOWIS             Types out current time of day: " The time is now hh:mm:ss"
        IO        Logical Unit for output:  6 for TT:, 7 for LP:.     I2   <=
          +       > time

NUTIME            Update real seconds by integer seconds, into time threesome.
        KHR       Integer hours.                                      I2    >
        KMI       Integer minutes.                                    I2    >
```

```
      KSE     Integer seconds.                                      I2   >
      CSEC    Real seconds to be updated for next interval.         R4  <>
      NSEC    Integer seconds as interval.                          I2  <=

 PGA          Reads A/D using selected PGA range.
      ICHAN   A/D Channel number (0-15).                            I2  <=
      IGAIN   Gain factor: 1, 2, 5 or 10.                           I2  <=
      IRAW    Raw A/D reading (0-4095).                             I2   >
       +      > ADREAD

 QPRINT       Prints out string using Fortran Formatting, without CRLF.
      STRING  Byte array to be printed out (terminated by 0 byte).   B*  <=
        !      TYPEs out STRING(I),I=1,LEN(STRING) to FORMAT('$',80A1)

 RANGE        Simply finds the minimum and maximum values of an array.
      N       Number of points (of interest) in the array.          I2  <=
      X       Array of values.                                      R*  <=
      XMIN    Minimum value of the array.                           R4   >
      XMAX    Maximum value of the array.                           R4   >

 RANGES       Finds the minimum and maximum values of the X and Y arrays.
      N       Number of (pairs of) points in the X and Y arrays.    I2  <=
      X       Array of X values.                                    R*  <=
      Y       Array of Y values.                                    R*  <=
      XMIN    Minimum value of X array.                             R4   >
      XMAX    Maximum value of X array.                             R4   >
      YMIN    Minimum value of Y array.                             R4   >
      YMAX    Maximum value of Y array.                             R4   >
        !     Values are not used to set the /VIEW/ COMMON.
        !     User may want to CALL SETS immediately after CALL RANGES.

 RESET        Grounds control signal DS0 momentarily.  RESETs EAA.  (MACRO)

 SDELAY       Waits for a specified number of seconds.
      SEC     Number of seconds for program to pause.               R4  <=
       +      > itwait, jafix, jjcvt
        !      Timing is accurate to nearest 1/60 second using ITWAIT.
        !      Needs FB monitor.  Completion routines continue to work.

=SECT4        Converts integer time foursome into real seconds.     R4   >
      ITIM    Integer time foursome (IH,IM,IS,IT)                   I8  <=

 SETEAA       Sets up COMMON for simple EAA data interpretation.
       /EAA/ EAADP(10),EAADPM(9),EAACON(9),EAASN(9),EAAVN(9)
        !     Subscript I refers to channel I-1 of EAA.
        !     /EAA/ holds the following data:
        !     EAADP - boundary diameters in microns.
        !     EAADPM - mean diameters in microns.
        !     EAACON - #/cc per volt (or per picoamp), sensitivity.
        !     EAASN - surface area per number (square microns each).
        !     EAAVN - volume per number (cubic microns each).
        !     This is the trivial inversion approach neglecting
        !      any EAA channel cross-sensitivity.  See EAASIZ.
 SETJSW         Sets or clears a selected bit in the Job Status Word.
        MBIT    Number of bit to be set (negative if to clear it).       I2  <=
                 Use 6 to inhibit terminal wait.
                 Use 12 for special input mode, each character available.
                 Use 14 to disable automatic conversion to upper case.
         +      > ipeek, ipoke

 SHOJSW         Types out Job Status Word pattern in Octal.
         +      > ipeek

 SPECIN         Waits for a single character in special input mode.
        CHAR    Single character which user just entered.                B1   >
        ECHO    Controls echo as follows:                                I2  <=
                 =0 no echo ; =1 echo ; =-1 echos any input except RETURN
         +      > ipeek, ipoke, ittinr, ittour

=SST3           Converts time threesome to short ASCII time hh:mm:ss.    A8   >
        KHR     Integer hours.                                           I2  <=
        KMI     Integer minutes.                                         I2  <=
        KSE     Integer seconds.                                         I2  <=

=SSTH           Converts real hours to short ASCII time, hh:mm:ss.       A8   >
        HOUR    Real hours.                                              R4  <=
         +      > SST3

=SSTS           Converts real seconds to short ASCII time, hh:mm:ss.     A8   >
        SEC     Real seconds.                                            R4  <=
         +      > SST3

 STAT           Calculates simple statistics for a real array.
        N       Number of values.                                        I2  <=
        VAL     Array of values.                                         R*  <=
        AV      Average of values.                                       R4   >
        SD      Standard deviation of values.                            R4   >
        SL      Slope of values, by linear regression. (per point)       R4   >
         +      > STATS

 STATAD         Add point to Mth summation set for statistics.
        M       Set number.  Currently may be 1 to 40.                   I2  <=
        X       Data value to add to Mth set.                            R4  <=
         /STAT/ N(40),S1(40),S2(40),SP(40)

 STATS          Calculates simple statistics given three summations.
        N       Number of (X) values.                                    I2  <=
        S1      Summation of (X).                                        R4  <=
        S2      Summation of (X squared).                                R4  <=
        SP      Summation of (X times i), where i denotes ith value.     R4  <=
        AV      Average of (X) values.                                   R4   >
        SD      Standard deviation (estimator) of (X) values.            R4   >
        SL      Slope of (X) values, by linear regression (per point).   R4   >

 STEP           Grounds DS3 control signal momentarily.  STEPs EAA.   (MACRO)
```

```
STINIT          Initialize M-th summation set for statistics.
        M       Number of summation set.  Currently may be 1 to 40.     I2  <=
        /STAT/ N(40),S1(40),S2(40),SP(40)}

T4ADD           Adds two time foursomes together.
        NTIME   Time as an integer foursome, which is to be updated.    I8  <>
        ITIME   Time interval as an integer foursome, added to NTIME.   I8  <=
          !     NTIME = NTIME + ITIME

TCHECK          Checks if internal time is within range given by real ticks.
        TINT    Internal Time of interest.                              J4  <=
        HMIN    Minimum Time in real hours.                             R4  <=
        HMAX    Maximum Time in real hours.                             R4  <=
        IF      Flag to tell if TINT is within range:                   I2   >
         .       0 if within range, -1 if low, +1 if high.
         +      > ajflt, jjcvt

TIME4           Makes integer time foursome from its expanded form.
        IH      Integer hours.                                          I2  <=
        IM      Integer minutes.                                        I2  <=
        IS      Integer seconds.                                        I2  <=
        IT      Integer ticks.                                          I2  <=
        ITIM    Integer time foursome, (IH,IM,IS,IT).                   I8   >
          !     Trivial subroutine.

TIME4X          Converts integer time foursome to its expanded form.
        ITIM    Integer time foursome, (IH,IM,IS,IT).                   I8  <=
        IH      Integer hours.                                          I2   >
        IM      Integer minutes.                                        I2   >
        IS      Integer seconds.                                        I2   >
        IT      Integer ticks.                                          I2   >
          !     Trivial subroutine.

TIMEIS          Types out " Elapsed time is hh:mm:ss at Time hh:mm:ss."
        HSTART  Initial or reference time in hours.                     R4  <=
        IO      Logical Unit for output:                                I2  <=
         +      > HRTINT, SSTH, gtim

TIMEX           Converts Internal Time into multiple forms stored in COMMONs.
        TINT    Internal Time.                                          J4  <=
         /TINT/ TINT1
         /JTIM/ JTIM
         /TICS/ TICS
         /SECS/ SECS
         /HOUR/ HOURS
         /ITIM/ IH,IM,IS,IT
         /TSTR/ TSTR(12)
         +      > jmov, jjcvt, ajflt, cvttim, transl

=TINTHR         Converts real hours into Internal Time.                 J4   >
        HRS     Real hours.                                             R4  <=
         +      jafix, jjcvt
```

```
=TINTST         Converts full time string into Internal Time.          J4   >
       TSTR     Time String, 'hh:mm:ss:tt', terminated with 0 byte.    B12 <=
        +       > jtime

 TREADY         Clears IOFLAG in /TINPUT/ so new terminal input can be used.
        /TINPUT/ IOFLAG
          !     This routine should be called at the end of a user
          !      KEYCOM routine so that KEYCHK can look at more terminal
          !      input and know that the prior input has been processed.

 UPCASE         Forces automatic conversion of terminal input to upper case.
        +       > SETJSW
         ! This is the normal RT-11 input mode.

 V132C          Sets VT100 family terminal to 132 characters screen width.
        +       > print

 V80C           Sets VT100 family terminal to 80 characters screen width.
        +       > print

 VAD            Reads A/D voltage once, using autoranging of PGA.
       ICHAN    A/D Channel Number [0-15 normally]                     I2  <=
       VOLTS    Voltage Reading from A/D [0.-10.]                      R4   >
        +       > PGA
         !      Not recommended!  A/D ranges do overlap perfectly.
         !      And Programmable Gain Amplifier does not significantly
         !       improve A/D resolution . . . use signal averaging
         !       routines instead.

 VADD           Adds points to the video screen plot.
       NN       Number of points to add.                               I2  <=
       X        Array of X values.                                     R*  <=
       Y        Array of Y values.                                     R*  <=
        /VIEW/
        +       > DSPLY, VOFF, VSETG0, VSETG1
         !      May set video attributes for points before CALL VADD
         !       by using VBLINK, VBOLD, etc., and VOFF afterwards.

 VAT            Sets video attributes for string.
       ATTRIB   List of Video Attributes in string form, where         B4  <=
        .        a 0 or space terminates the list (up to 4),
        .        and the following attributes may be used:
        .               'B' or 'b' = Bold
        .               'F' or 'f' = Flash (blink)
        .               'R' or 'r' = Reverse video
        .               'U' or 'u' = Underline
        +       > VBLINK, VBOLD, VOFF, VREV, VUNDER

 VBLINK         Causes future VT output characters to blink (Flash).
        +       > print

 VBORD          Generates the Border of a plot (directly to the VT).
```

```
        /VIEW/ LMIN,LMAX,IMIN,IMAX
        /VAXES/ LMAR,IMAR
        +       > VHLINE, VSETG0, VSETG1, VVLINE, scopy

VBOLD           Causes future VT output characters to be in Boldface (bright).
        +       > print

VCHSET          Establishes standard character sets on the VT100 terminal.
        +       > print
         ! Character Set G0 is American and Set G1 is Special Graphics.

VCLEAR          Simply clears the video screen.
        +       > print

VCRES           Restores Cursor Attributes such as position (from last VCSAV).
        +       > print

VCSAV           Saves Cursor Attributes such as position (for VCRES later).
        +       > print

VDAC0           Sets Digitial-to-Analog Converter #0 to selected voltage.
       VOLTS    Output Voltage at DAC0.  Normally -10. to +10. volts.   R4   <=
        +       > DAC0

VDAC1           Sets Digitial-to-Analog Converter #1 to selected voltage.
       VOLTS    Output Voltage at DAC1.  Normally -10. to +10. volts.   R4   <=
        +       > DAC1

VDOWN           Moves video cursor down a selected number of positions.
       N        Number of positions to move down (0-23).                I2   <=
        +       > print

VHLINE          Draws a horizontal line onto the VT100, using graphics.
       LINE     Line number (1-24) for entire line.                     I2   <=
       FORM     Character Pattern: Start/Rest/End, or Ends/Rest or All  B4   <=
       ICOL     Starting column number for line (1-132).                I2   <=
       JCOL     Ending column number for line (1-132).                  I2   <=
        +       > VPUT

VHOME           Simply moves the video cursor home (top left of screen).
        +       > print

VINIT           Initializes the video terminal, with following CALLs:
        +       > VHOME, VCLEAR, VOFF, VCHSET, VSETG0, VSCROL(1,24)
        !       Note: terminal line width is not affected.

VLED            Controls terminal LEDs (1 on VT100s, 4 on ADM36 keyboard).
       LED      Use 1-4 to turn selected LED on; 0 turns all off.       I2   <=
        +       > print

VLEFT           Moves video cursor left a selected number of positions.
       N        Number of positions to move left (0-131).               I2   <=
        +       > print
```

```
VOFF            Causes future VT output characters to have normal attributes.
       +        > print
        !        VOFF cancels VBOLD, VBLINK, VREV, and VUNDER calls.

VPAUSE          Waits at bottom of plot for user to enter command character.
      CHAR      Character which user enters.                          B1   >
       +        > SPECIN, VPRINT, VPUT, print
        !       Routine automatically prints "Pause:" and will acknowledge
        !        acknowledge user input.

VPRINT          Prints a character string, suppressing the carriage return.
      STRING    Character string, terminated by a zero byte.          I2  <=
       +        > print
        !       Uses CALL PRINT, and the RT-11 String Handling package;
        !        String must terminate with a zero byte (or a 128. byte,
        !        in which case CALL PRINT would work identically well).

VPUT            Move Video Cursor to selected position.
      LINE      Line Number (1-24) ; sets vertical position (Y).      I2  <=
      ICOL      Column Number (1-132) ; sets horizontal position (X). I2  <=
       +        > print, transl

VPUTST          Puts a string onto VT100 at selected location.
      LINE      Line Number at which string begins (1-24).            I2  <=
      ICOL      Column Number at which string begins (1-132).         I2  <=
      STRING    Character String, terminated by a 0 byte.             B*  <=
       +        > VPRINT, VPUT
        !       VPUTST leaves cursor at end of string.

VREAD           Read selected A/D Channel once, returning voltage.
      ICHAN     A/D Channel Number [0-15 normally]                    I2  <=
      VOLTS     Voltage Reading from A/D [0.-10.]                     R4   >
       /ADCALO/ VCON
       +        > ADREAD

VREADM          Reads selected A/D Channel repeatedly with millisecond waits.
      ICHAN     A/D Channel Number [0-15 normally]                    I2  <=
      IRAW      Array for Raw A/D Readings                            I*   >
      KOUNT     Number of A/D Readings to take                        I2  <=
      VAV       Average A/D Voltage                                   R4   >
      VSD       Standard Deviation of Voltage Signal                  R4   >
      VSL       Rate of Change of Signal (Volts/Second)               R4   >
      SECS      Seconds Required for Sampling                         R4   >
      MILSEC    Milliseconds to delay between A/D Reads               I2  <=
       /ADCALO/ VCON
       +        ADREAD, ELAPSE, ISTAT, MDELAY, gtim

VREADS          Reads selected A/D Channel repeatedly and quickly.
      ICHAN     A/D Channel Number [0-15 normally]                    I2  <=
      IRAW      Array for Raw A/D Readings                            I*   >
      KOUNT     Number of A/D Readings to take                        I2  <=
      VAV       Average A/D Voltage                                   R4   >
```

```
        VSD      Standard Deviation of Voltage Signal                R4    >
        VSL      Rate of Change of Signal (Volts/Second)             R4    >
        SECS     Seconds Required for Sampling                       R4    >
         /ADCALO/ VCON
         +       > ADSET, ELAPSE, ISTAT, gtim

VREADW           Reads selected A/D Channel at system clock intervals.
        ICHAN    A/D Channel Number [0-15 normally]                  I2   <=
        IRAW     Array for Raw A/D Readings                          I*    >
        KOUNT    Number of A/D Readings to take                      I2   <=
        VAV      Average A/D Voltage                                 R4    >
        VSD      Standard Deviation of Voltage Signal                R4    >
        VSL      Rate of Change of Signal (Volts/Second)             R4    >
        SECS     Seconds Required for Sampling                       R4    >
        IH       Hours  . . .                                        I2   <=
        IM       Minutes . . .                                       I2   <=
        IS       Seconds . . .                                       I2   <=
        IT       Ticks, delay time between A/D Reads.                I2   <=
         /ADCALO/ VCON
         +       > ADREAD, ELAPSE, ISTAT, gtim, isleep

VREADX           Repeatedly reads A/D channel and then does statistics.
        ICHAN    A/D Channel Number [0-15 normally]                  I2   <=
        IRAW     Array for Raw A/D Readings                          I*    >
        KOUNT    Number of A/D Readings to take                      I2   <=
        VAV      Average A/D Voltage                                 R4    >
        VSD      Standard Deviation of Voltage Signal                R4    >
        VSL      Rate of Change of Signal (Volts/Second)             R4    >
        SECS     Seconds Required for Sampling                       R4    >
         /ADCALO/ VCON
         +       > ADREAD, ELAPSE, ISTAT, gtim
          !      This routine can be customized several ways since it
          !       serves as a model program.  Currently it is like
          !       VREADS except slower (but still as fast as can be).

VRESET           Resets video terminal to power-up state.
         +       > print

VRIGHT           Moves video cursor right a selected number of positions.
        N        Number of positions to move right (0-131).          I2   <=
         +       > print

VREV             Causes future VT output characters to be reverse video mode.
         +       > print

VSCAN            Flexible multichannel A/D Sampler with Statistics.
        NCHAN    Number of Channels (if <=0, ICHAN is 16 on/off values)  I2   <=
        KOUNT    Number of A/D Reads to take on each channel.        I2   <=
        MODSAM   Sampling Mode [0=constant sums, 1=sum afterwards]   I2   <=
        MODDEL   Delay Mode [0=none, 1=MILSEC, 2=ITIM(4), 3=DELSEC]  I2   <=
        ICHAN    List of A/D Channels [NCHAN or 16 elements]         I*   <=
        IRAW     Array to hold Raw A/D Readings (iread,jchan)        I*    >
        VAV      Average A/D Voltage Array                           R*    >
        VSD      Standard Deviation Array of Voltage Signal                R*   >
        VSL      Rate of Change of Signal Array (Volts/Second)             R*   >
        SECS     Seconds Required for Sampling                             R4   >
        XTIM     Timing parameter, depends on MODDEL value:                ?    <=
                  0 ignored, 1 for MILSEC, 2 for ITIM(4), 3 for seconds
         +       > ADREAD, DOSTAT, ELAPSE, ISTAT, MDELAY, SDELAY
         +       > STATAD, STINIT, gtim, isleep
          !       Note that the array allocation depends on NCHAN;
          !        if NCHAN positive, exactly NCHAN channels are used;
          !        otherwise full 16 channel storage is used, and
          !        ICHAN is not channel numbers but 1 for on or 0 for off.
          !       The value of MODSAM may effect the timing of the program as
          !        well as whether the IRAW array is used (not if MODSAM=0).
          !        Use MODSAM=0 to save space and sample continuously.
          !        Use MODSAM=1 for very rapid A/D sampling (with MODDEL=0)
          !        or to save all the A/D readings taken.
          !       SECS is always the time between first and last A/D read;
          !        if MODSAM=1 the routine may take much longer than SECS
          !        to finish.

VSCROL           Sets scolling region of video screen (at least two lines).
        MIN      Starting Line Number (1-23).                              I2   <=
        MAX      Ending Line Number (2-24).                                I2   <=
         +       > print
          !      Normal is CALL VSCROL(1,24).  Remember to restore to
          !       normal at end of program if special scrolling was used.

VSETE            Sets extent of screen to be used for plotting.
        LIMIN    Starting Line Number for plot region; 2 is standard.      I2   <=
        LIMAX    Ending Line Number for plot region; 22 is standard.       I2   <=
        ICMIN    Starting Column Number for plot region; 21 is standard.   I2   <=
        ICMAX    Ending Column Number for plot region; 121 is standard.    I2   <=
         /VIEW/  LMIN,LMAX,IMIN,IMAX
          !      Standard values get used if VSETE is not called.

VSETG0           Use Standard Character Set G0 (Normally US ASCII) on VT100.
         +       > print

VSETG1           Use Alternate Character Set G1 (Normally Graphics) on VT100.
         +       > print

VSETS            Sets range of plot in user units.
        YLO      Minimum (bottom) value of the Y variable.                 R4   <=
        YHI      Maximum (top) value of the Y variable.                    R4   <=
        XLO      Minimum (left) value of the X variable.                   R4   <=
        XHI      Maximum (right) value of the X variable.                  R4   <=
        IARG     Form of X labels: 0 normal, 1 for hr:mi, 2 for elapsed    I2   <=
                  time in hr:mi or mi:se or se:ti.
         /VIEW/
          !      This routine must be called before plotting.  It also
          !       initializes COMMON for plot region and tick positions.

VSETT            Sets number and spacing of ticks and border.
```

```
        NYTCKS  Number of Y-axis ticks; 5 is standard.              I2  <=
        NXTCKS  Number of X-axis ticks; 5 is standard.              I2  <=
        NYMAR   Number of spaces for Y border offset; 1 is standard. I2 <=
        NXMAR   Number of spaces for X border offset; 9 is standard. I2 <=
         /VAXES/ LMAR,IMAR,LTICKS,ITICKS
          !     Just puts values into /VAXES/ COMMON.

VSHIFT          Moves SCREEN plot view one division (tick) to the right.
         /VIEW/ LMIN,LMAX,IMIN,IMAX,YMIN,YMAX,XMIN,XMAX
         /VAXES/ LMAR,IMAR,LTICKS,ITICKS
         /VS/ SCREEN(133,24)
          !     This only affects the SCREEN matrix.  Use VXPLOT(0)
          !      to display the shifted plot on the VT100.

VSINIT          Initializes the SCREEN matrix for plotting.
        IARG    Positive to also clear the VT100 terminal.          I2  <=
         /VS/ SCREEN(133,24)
         +      > V132C, VINIT

VSTR            Advanced Video String Placement Routine.
        STRING  Character string to be displayed, terminated by 0 byte. B*  <=
        LINE    Line where the string goes (1-24).                  I2  <=
        ICOL    Column where the string begins (1-132).             I2  <=
        ATTRIB  Video Attributes, selected among ('B','F','U','R')  A4  <=
        NEXT    Sets resulting cursor position:                     I2  <=
                 0=home, 1=end of string, 2=next line, -1=last spot
         +      > VAT, VCRES, VCSAV, VOFF, VPRINT, VPUT, print
          !     This routine should serve all VT100 string placement
          !       needs except where string is not simply left to right.
          !     See VAT for more info on ATTRIB, which are
          !      Bold, Flashing, Underline, and Reverse.

VTICKS          Displays ticks and labels on the video screen plot.
        IARG    Let IARG=1 for tick labels, IARG=0 for no tick labels.  I2  <=
         +      > DSPLY, VPUTST, VSETG1, VSETG0

VTITLE          Displays the pre-selected title onto the VT100.
        LINE    Line onto which title is to be placed.  (0 gives first) I2  <=
        ATTRIB  Video attribute string ('BRFU'; see VAD) for title.  B4  <=
         /VIEW/
         /TITLE/
         +      > VSTR

VUNDER          Causes future VT output characters to be underlined.
         +      > print

VUP             Moves video cursor up a selected number of positions.
        N       Number of positions to move up (0-23).              I2  <=
         +      > print

VVLINE          Draws vertical line onto video screen, using graphics.
        ICOL    Column number (1-24) for entire line.               I2  <=
        FORM    Character Pattern: Start/Rest/End, or Ends/Rest or All  B4  <=
        LINE1   Starting Line Number (1-24).                        I2   <=
        LINE2   Ending Line Number (1-24).                          I2   <=
         +      > VDOWN, VLEFT, VPUT

VXADD           Adds data points to a VT100 plot (VXPLOT).
        N       Number of points to add.
        X       Array of independent variable values.
        Y       Matching array of dependent variable values.
         /VIEW/ LMIN,LMAX,IMIN,IMAX,YMIN,YMAX,XMIN,XMAX
         /VS/ SCREEN(133,24)
         +      > DSPLY, VBLINK, VBOLD, VOFF, VSETG0, VSETG1

VXPLOT          Generates a plot of X-Y data onto the video screen.
        N       Number of points (if positive), or:                 I2   <=
         .       0=empty plot, -1=redraw, -2=empty saving SCREEN.
        X       Array of independent variable values.               R*   <=
        Y       Array of dependent variable values.                 R*   <=
         /VIEW/ LMIN,LMAX,IMIN,IMAX
         /VAXES/ LMAR,IMAR,LTICKS,ITICKS,ISTIME
         +      > V132C, VBORD, VHOME, VINIT, VSINIT, VTICKS
         +      > VTITLE, VXADD, VXSHOW
          !     This routine both plots directly to VT100 screen
          !      and saves data in the SCREEN image matrix.
          !     VSETE must be called before VXPLOT, otherwise this
          !      is basically a stand-alone plotting routine.
          !     VXPLOT always clears VT100 screen and draws axes & ticks.
          !     N<0 will save the old SCREEN matrix.

VXPLT1          Generates a plot of equispaced data onto the VT100.
        N       Number of points (if positive), or:                 I2   <=
         .       0=empty plot, -1=redraw, -2=empty saving SCREEN.
        Y       Array of dependent variable values.                 R*   <=
        XMIN    Minimum (starting) value of independent variable.   R4   <=
        XMAX    Maximum (ending) value of independent variable.     R4   <=
         /VIEW/ LMIN,LMAX,IMIN,IMAX
         /VAXES/ LMAR,IMAR,LTICKS,ITICKS,ISTIME
         +      > V132C, VBORD, VHOME, VINIT, VSINIT, VTICKS
         +      > VTITLE, VXADD, VXSHOW
          !     This is a single dimensional version of VXPLOT,
          !      where the independent X data is equispaced.

VXSHOW          Redraws active window on video plot from SCREEN matrix.
         /VIEW/ LMIN,LMAX,IMIN,IMAX,YMIN,YMAX,XMIN,XMAX
         /VAXES/ LMAR,IMAR,LTICKS,ITICKS,ISTIME
         /VS/ SCREEN(133,24)
         +      > VSETG0, VSETG1, VSTR
          !     Works with the VXPLOT and VXADD routines.

WHEN            Internal Time minus Real Ticks into Elapsed Time String.
        TINT    Internal Time.                                      J4   <=
        START   Starting time in real ticks.                        R4   <=
        TSTR    Full Time String, 'hh:mm:ss:tt', terminated by 0 byte.  B12  >
         +      > CVT4ST, ajflt, cvttim, jafix, jjcvt, jmov
```

```
                       !     This routine is not particularly recommended, since
                       !      its units of real ticks represent a proliferation of
                       !      the units used for time of day.  It should probably
                       !      be replaced by HRTINT or ELAPSE, with CVHRST or SSTS.

---------------------------------------------------------------------------

!       Note that the final entries for each subroutine argument are the
!        variable type and length (such as R4 for REAL*4 or A12 for a
!        twelve-element BYTE array being used as a character string),
!        and a visual code for when the value has meaning, i.e.:
!               <= if variable needed on routine entry and unchanged
!               <> if variable needed on routine entry and changed
!                > if variable defined only on return from routine
!       The + entries (with > also) denote other RTLIB (uppercase)
!        or Syslib (lowercase) routines that are directly called by the
!        given RTLIB routine.  See Programmers' Reference Manual for
!        information on (FORTRAN) SYSLIB routines.
```

```
C**********************          TLIB -- Time Handling Library
C
C       This Documentation Written on 8-NOV-84 by DRW
C
C***                     TIME FORMATS
C
C                Times may be expressed in the following formats:
C
C A)    1.  TINT - Internal Time, swapped-word INTEGER*4 Ticks          *4
C       2.  JTIM - INTEGER*4 Ticks (since midnight)                     *4
C B)    3.  TICS - REAL*4 Ticks (a clock tick is 1/60 second)           *4
C       4.  SECS - REAL*4 Seconds                                       *4
C       5.  HRS  - REAL*4 Hours                                         *4
C C)    6.  ITIM - Four element INTEGER*2 Array (see 7)                 *8
C       7.  IH,IM,IS,IT - Foursome of INTEGER*2 values                  *8
C D)    8.  SST  - Eight element byte array, as 'hh:mm:ss'              *8
C       9.  TSTR - Twelve byte character string, 'hh:mm:ss:tt'//0      *12
C
C
C A)    TINT and JTIM are doubleword tick counts.
C B)    TICS, SECS, and HRS are floating point times in the specified units.
C C)    ITIM(1:4) and IH,IM,IS,IT are integer foursomes holding time.
C D)    SST and TSTR are ASCII time representations.
C
C       Note:   The above variable nomenclature is used throughout my
C               documentation as well as within the timing programs -- DRW
C
C***                     TIME INTERCONVERSION
C
C       Intraconversion within any mode of expression is simple:
C
C A)    CALL JJCVT(TINT)                TINT -> JTIM
C       CALL JJCVT(JTIM)                JTIM -> TINT
C
C B)    TICS=60.*SECS                   SECS -> TICS
C       HRS=TICS/216000.                TICS -> HRS
C        (and so forth, as TICS, SECS, HRS are trivial to interconvert)
C
C C)    EQUIVALENCE (ITIM(1),IH),(ITIM(2),IM),(ITIM(3),IS),(ITIM(4),IT)
C       Or might interchange these between routines by COMMONs,
C        or simply by direct assignments (i.e., =).
C       Take care to use the right number of names as subroutine arguments!
C
C D)    SST is a shortened time string which begins the same as TSTR.
C       SST is 'hh:mm:ss' without a final null byte for string termination.
C       TSTR is 'hh:mm:ss:tt' with a terminating byte so it is
C        recognized by the system as a character string.
C
C       The following interconversion paths are available as
C       system requests:
C
```

```
C        CALL CVTTIM(TINT,IHR,IMIN,ISEC,ITIC)     TINT -> IH,IM,IS,IT
C        CALL JTIME(IHR,IMIN,ISEC,ITIC,JT)        IH,IM,IS,IT -> TINT
C        CALL TIMASC(TINT,SST)                    TINT -> SST
C
C        TICS = AJFLT(JTIM)                       JTIM -> TICS
C        ierr = JAFIX(TICS,JTIM)                  TICS -> JTIM
C
C
C        The paths already shown allow everything except for going from
C        a time string to something else, and for that case the time string
C        could be decoded to an integer foursome, thus closing the matrix
C        with no isolated nodes where "you can't get there from here."
C        Still, figuring out and encoding the appropriate time conversion
C        pathways might be a bothersome chore for the user.  Hence, to keep
C        the user's code short and simple, and to minimize the chance
C        of careless errors and lost precision, TIMLIB contains several
C        direct conversion subroutines.  Here is the current set of them:
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C              SUMMARY OF TIME INTERCONVERSION SUBROUTINES
C
         TINT     JTIM     TICS     SECS     HRS      ITIM     IH-IT    SST      TSTR
         ....     ....     ....     ....     ...      ....     .....    ...      ....
TINT :   [JMOV]   JJCVT    (timex)  (timex)  hrtint   (timex)  CVTTIM   TIMASC   when
JTIM :   JJCVT    [JMOV]   AJFLT     -        -        -        -        -        -
TICS :    -       JAFIX     [=]     /60.     /2.16E5   -        -        -        -
SECS :    -        -       *60.      [=]     /3600.    -        -       ssts      -
HRS  :   tinthr    -       *2.16E5  *3600.    [=]     cvhrt4    -       ssth     cvhrst
ITIM :    -        -        -       sect4    hrt4      [=]     time4x    -       cvt4st
IH-IT:   JTIME     -        -        -       hrt4x    time4     [=]     sst3      -
SST  :    -        -        -        -       hrsst    DECODE   DECODE    [=]      =
TSTR :   tintst    -        -        -       hrstr    DECODE   DECODE     =      [SCOPY]

         First Column is starting form, Top Row is final form.
         System subroutines are capitalized, TLIB routines are in lower case.
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C***                 ALPHABETICAL LISTING OF TIME LIBRARY ROUTINES:
C
C
C        CALL ASKTIM(ITIM,ICHECK)         Asks for ITIM   Confirms if ICHECK<>0.
C
C +      CALL CVHRST(HRS,TSTR)            HRS --> TSTR
C
C +      CALL CVHRT4(HRS,ITIM)            HRS --> ITIM
C
C +      CALL CVT4ST(ITIM,TSTR)           ITIM --> TSTR
C
C S+     CALL CVTTIM(TINT,IH,IM,IS,IT)    TINT --> IH,IM,IS,IT
C
C        = ELAPSE(TINT1,TINT2)            TINT2 - TINT1 --> SECS
```

```
C
C S*    CALL GTIM(TINT)                 clock time --> TINT
C
C +      = HRSST(SST)                   SST --> HRS
C
C +      = HRSTR(TSTR)                  TSTR --> HRS
C
C +      = HRT4(ITIM)                   ITIM --> HRS
C
C +      = HRT4X(IH,IM,IS,IT)           IH,IM,IS,IT --> HRS
C
C +      = HRTINT(TINT)                 TINT --> HRS
C
C S*    CALL ICMKT(ID,TINT)             Cancel completion routine having
C                                       id ID (0 for all).  TINT is time left.
C
C S*    CALL ISCHED(IH,IM,IS,IT,AREA8,ID,fcrt)  Schedule Fortran completion
C                                       routine fcrt at time IH,IM,IS,IT
C                                       with id ID and 8-byte linkage AREA8.
C
C S*    CALL ISLEEP(IH,IM,IS,IT)        Sleep for indicated time interval.
C
C S*    CALL ITIMER(IH,IM,IS,IT,AREA8,ID,fcrt)  Schedule Fortran completion
C                                       routine fcrt after time interval
C                                       IH,IM,IS,IT passes, with id ID.
C
C S*    CALL ITWAIT(TINT)               Sleep for indicated time interval.
C
C S*    CALL IUNTIL(IH,IM,IS,IT)        Sleep until indicated time arrives.
C
C S+    CALL JTIME(IH,IM,IS,IT,TINT)    IH,IM,IS,IT --> TINT
C
C S*    CALL MRKT(ID,acrt,TINT)         Schedule completion routine acrt
C                                       with id ID after interval TINT.
C
C *     CALL NEWTIM             Returns current time in multiple COMMONs
C
C *     CALL NOWIS(IO)          Prints current time 'hh:mm:ss' at unit IO.
C
C       CALL NUTIME(IH,IM,IS,SEC,NSEC)  SEC + NSEC --> SEC ; IH,IM,IS
C
C *     CALL SDELAY(SEC)        Waits specified number of seconds.  Needs F/B.
C
C S*     = SECNDS(SECS0)                Clock Time - SECS0 --> SECS (rounded)
C                                        Note: Use NEWTIM for higher accuracy.
C
C +      = SECT4(ITIM)                  ITIM --> SECS
C
C +      = SST3(IH,IM,IS)               IH,IM,IS --> SST
C
C +      = SSTS(SECS)                   SECS --> SST
C
C +      = SSTH(HRS)                    HRS --> SST
```

```
C
C       CALL T4ADD(NTIME,ITIME)         NTIME(1:4) + ITIME(1:4) --> NTIME(1:4)
C
C       CALL TCHECK(TINT,TMIN,TMAX,IF)  Checks if TINT is below, within, or
C                               above the time interval given in tics by
C                               TMIN thru TMAX; IF is -1,0,1 respectively.
C
C S+    CALL TIMASC(TINT,SST)           TINT --> SST
C
C S*    CALL TIME(SST)                  clock time --> SST
C
C       CALL TIME4(IH,IM,IS,IT,ITIM)    IH,IM,IS,IT --> ITIM
C
C       CALL TIME4X(ITIM,IH,IM,IS,IT)   ITIM --> IH,IM,IS,IT
C
C *     CALL TIMEIS(TSTART,IO)  Prints current and elapsed time at unit IO.
C                               TSTART is REAL*4 Ticks.
C
C +     CALL TIMEX(TINT)                TINT --> all other formats via COMMON
C
C +     = TINTHR(HRS)                   HRS --> TINT
C
C +     = TINTST(TSTR)                  TSTR --> TINT
C
C       CALL WHEN(TINT,TICS0,TSTR)      TINT - TICS0 --> TSTR
C
C
C S  denotes SYSLIB system routine.
C *  denotes a real time routine.
C +  denotes a routine, mentioned earlier, which just does time conversion.
C
C       The following labeled COMMONS are used by TIMEX and NEWTIM, and
C       should be included in the user's code, as needed:
C
C       COMMON /TINT/ TINT              ! Internal Time
C       COMMON /JTIM/ JTIM              ! INTEGER*4 Tics
C       COMMON /TICS/ TICS              ! REAL*4 Tics
C       COMMON /SECS/ SECS              ! REAL*4 Seconds
C       COMMON /HOUR/ HOURS             ! REAL*4 Hours
C       COMMON /ITIM/ IH,IM,IS,IT       ! INTEGER*2 Time Foursome
C       COMMON /TSTR/ TSTR              ! BYTE Array (12), 'hh:mm:ss:tt'
C
C***                    FOR FURTHER INFORMATION:
C
C               Please see the RT-11 Programmer's Reference Manual,
C       Version 4, as well as the TLIB Fortran Source Codes kept on the
C       Subroutine Library Disks, in files (TIMLIB,T2LIB,T3LIB).FOR.
C
C       Note: All the TLIB subroutines have been incorporated
C             into RTLIB.  Refer to RTLIB.DOC for more information.
```

```
---------------------- NEW RT-11 PROGRAM STORAGE ----------------------

 Source Disks:  RTLIB   WATCH   SAMPLE  SUNDRY  DOC

 System Disks:  FULL    RTLIB   WATCH   SAMPLE  BASIC   MINIMUM

  ==>   Look for the green dot; that's the new working version!

        Note:  The Roof Lab RT-11 has single density RX01 disk drives,
               so not as much can be placed on a diskette, thus,
               FULL is split into FORTRAN, MACRO, and UTILITY;
               SAMPLE is split into AERSAM and SAMPLE;
               and RTLIB will not fit on a system disk with the
               needed LINKER and SYSLIB, so must be used in Keck 210.

--------------------------- DRW Nov-85 ------------------------------

        *** RTLIB Source ***    VT100 video, A/D, Timing, Stats Library

.FOR:   V       VSTR    VE      VI      VSX     VPX     VX      <VXLIB>
        VREAD   VDAC    VREADS  VSCAN                           <ADLIB>
        TIMLIB                                                  <TLIB>
        STAT    ISORT                                           <SLIB>
        CLOCK   KEYCHK                                          <WLIB>
        EAALIB  EAACOM                                          <EAALIB>
        VS      VP      DSTAT                           <VSLIB,VPLIB,DSLIB>
        CYCLE   VOLD    WHEN                                    (old)

.MAC    ADREAD  NEWEAA  MSWAIT  DAC     ADSET   ADSET2
        MSWAIT  MDELAY

.COM    RTLIB   VXLIB   ADLIB   TLIB    SLIB    WLIB    EAALIB  RTLIB1
        VSLIB   VPLIB   VXLIB2  DSLIB   VALLIB

.OBJ    VLIB    VXLIB   ADLIB   TLIB    SLIB    WLIB    EAALIB  RTLIB
        (These are object libraries.  RTLIB includes ALL the others.
         VSLIB & VPLIB, which plot to a screen image array and directly
         to the screen, respectively, have not been used since Dec-84,
         a fate shared by DSLIB, which is SLIB in Double Precision.
         VXLIB, a hybrid of VSLIB and VPLIB, is used for VT100 video.

        See the long RTLIB.DOC printout for a full description.

---------------------------------------------------------------------

        *** WATCH Source ***    Multichannel Data Acquisition Program

.FOR    WATCH   GREET   WOPEN   WSETUP  ADUSE   WWHEN   WLABEL  VWINIT
        VWAIT   WSTART  WEND    SUMMRY  NOTE    DONOTE  NUINFO  KEYCOM
        WHELP   REFRSH  PROMPT  UPDATE  CHARAN  SHORAN  TUPDAT  FAD
        DOWCR   REVU    SLABEL
        DATINI  DATCMP  FILES
        REVIEW

.MAC    none (but uses some in RTLIB).

.COM    FWATCH  OWATCH  DATCMP

.OBJ    as needed.

.SAV    DATINI  WATCH   DATCMP  REVIEW

        WATCH.RUN

----------------------------------------------------------------------

        *** SAMPLE Source ***   Sampling Programs including DOEAA

.FOR    DOEAA   DO2EAA  SAVEAA  VTEAA   HISTOG
        SAVOPC
        SAVTOL
        ADPLOT  ADSCAN  VRTEST
        DASET   DASCAN

.COM    same as above
.SAV as space permits

----------------------------------------------------------------------

        *** SUNDRY Source ***   Assorted & Obsolete Programs

.FOR    ADFAKE  ADFCR   ADMAIN  ADSEEO  ADSTST  ADSAMP  ADTEST
        ASTEST  ATDCAL  BCDTST  CARB    CLKCAL  CLKTST  CLTEST
        COMTES  CONTRL  CRASH   CURTES  DATEST  DOEAA1  ECONT
        ESAMP   FILES   FITEST  GREAD   GSAMP   GSTEST  INTEST
        MELSAM  METLER  MSAMP   OPSAMP  OPTEST  PGACAL  RANTST
        RTEST   SHOEAA  SHOWST  STEST3  STEST4  STTEST  TDSPLY
        TESASK  TESPRI  TFORM   THERM   TPAUSE  TSLOPE  TV
        VPAUSE  VPTP    VPTS    VPTX    VRTEST

.COM    most of the above

.MAC    ADCAL   BSERV   CLOCK   CONSOL  CONO    CON1    CON2
        CON3    DOOPC   DRVTST  ECLOCK  MCLOCK  OPC1    OPC1L
        OPC2    OPC2L   SAMP    UNCODE

======================================================================

        *** FULL System ***     FORTRAN, MACRO, LINKER, Utilities

 * SWAP   .SYS  25       Extra space needed by system to stash info.
 * RT11FB.SYS  80        The Foreground-Background v4 Monitor.
 * TT     .SYS   2       Terminal Device Handler.
 * DY     .SYS   4       RX02 Double Density Disk Handler.
 + DW     .SYS   2       Decwriter Device Handler. (Modified LS.SYS)
```

```
  NL    .SYS   2      Null Device Handler.
  LS    .SYS   2      Serial Lineprinter Handler.
* PIP   .SAV  23      Peripheral Interchange Program.        .COPY,
* DUP   .SAV  41      File Duplication Program.              .COPY,
* DIR   .SAV  17      Directory Program.                     .DIR
+ KED   .SAV  60      VT100 Screen Editor, subset of EDT.    .EDIT
  LINK  .SAV  41      Linker.                                .LINK
  FORTRA.SAV 177      FORTRAN IV Compiler.                   .FORT
  FORMAT.SAV  19      Disketter Formatter.                   .FORMAT
  RESORC.SAV  15      System Resources Program.              .SHOW
  LIBR  .SAV  22      Library Program.                       .LIB
  MACRO .SAV  51      MACRO Assembly Language Compiler.      .MACRO
  BINCOM.SAV  10      Binary Comparison Program.
  DUMP  .SAV   8      Octal/ASCII File Dump Program.         .DUMP
  SYSLIB.OBJ 202      System and Fortran Object Library.
  CREF  .SAV   6      Cross-reference generator for LINK.
  SYSMAC.SML  42      System MACRO Library.
  SRCCOM.SAV  13      Source Comparison Program.
  SLP   .SAV   9      Code patch generator.
+ STARTF.COM   2      Start-up command file.
+ STARTF.TXT   1      Message printed by STARTF command file.
    Total    876/974 Blocks (512 bytes) on SS,DD 8.5" diskette.
```

* Denotes Files Essential to all System Disks.
+ Denotes Files recommended for all System Disks.
System Disks may be interchanged when the system is idle provided that the .SYS files are at the same location on both system disks. (All the Standard System Disks -- green dot -- are interchangable.)

APPENDIX D:

LISTINGS OF DATA ANALYSIS CODES

The following pages contain documentatation and source listings for the programs used to analyze EAA and OPC smog chamber aerosol data. These programs were written in Microsoft Fortran v3.20 running on the IBM AT and XT personal computers. After documentation describing the EAA data flowstream from acquisition to plotting, and a description of the key THREATS (THReasholding Eaa Analysis with Twomey Smoothing) program based mainly on Greg Markowski's ideas, are listings of the the following codes:

(1) ZEAA and EAALIB, programs for generating plottable files of EAA histogram size distributions;
(2) EAAINT, a program for time interpolation of EAA currents to correct for the asynchronous sampling of each channel within a cycle;
(3) THREATS, the EAA inversion code to correct for channel cross-sensitivity and provide smooth size distributions;
(4) SELECT, a program which selects and time averages the inverted and histogram distributions for plotting purposes;
(5) XOPC, a program for compressing OPC data and combining diluted and undiluted datasets;
(6) HISTOPC, a program for generating OPC histograms for plotting; and
(7) OPCIN, a version of Jim Crump's CINVERSE for automatically (using search techniques to find the proper smoothing parameter) invert sets of OPC data.

*** EAA DATA ACQUISITION, ANALYSIS, AND PLOTTING FILES ***

1. Take EAA data by the PDP-11 computer (in 210 Keck or Roof Lab) using either DOEAA (single EAA) or DO2EAA (dual EAA) RT-11 programs for EAA control, single averaging, and recording.

2. If the long form of data storage from DOEAA or DO2EAA was used (including standard deviations and trends for each EAA current reading), use the RT-11 program SAVEAA to compress the file to a standard EAA ASCII file, *.EA#,

   HOURS, (CURR(K),K=1,10).

3. If desired, interpolate the EAA data to compensate for asynchronous sampling of each channel within a dataset, by using EAAINT on either the VAX or an IBM PC/XT/AT. The resulting file *.EI# file has the same format as the *.EA#.

4. The ZEAA program can now be used on the VAX or IBM to prepare to plot channel profiles with time, assuming simple histogram data interpretation. ZEAA reads the ASCII *.EA# or *.EI# files and produces the unformatted *.ZE# and *.ZI# files, which may be ZPLOTTED using various PDL files as are described in EPLOT.DOC.

5. The THREATS program on the IBM-AT (evolved from Greg Markowski's SMoothed TWOMey algorithm, SMTWOM) will invert the *.EA# or *.EI# EAA data files, producing a smoothed distribution which would yield the observed currents to within a specified tolerance (default is 5%). An inversion technique is needed to compensate for cross-sensitivity between size-channels. The following output files are created by THREATS (THReasholding Eaa Analysis with Twomey Smoothing):
   a) an unformatted *.ZC file, which contains the time, channel currents, and inverted sub-currents (size distribution, in units of current, at 16 points per decade of particle diameter), for each EAA dataset. This is the primary output file; the others may be created later as needed.
   b) DIST.## and HIST.## are unformated size distribution files suitable for ZPLOTTING. E.g., DIST.1 contains the inverted number and volume distribution at time #1, while HIST.12 contains the histogram number and volume distributions at the 12th recorded time.
   c) *.ZN# and *.ZV# are the unformatted inverted Number and Volume profiles, respectively. They are suitable for ZPLOTTING.

   See THREATS.DOC for more information.

6. The SELECT program on the IBM-AT will read the *.ZC files created by THREATS and regenerate the *.ZV or *.ZN files. More significantly, it will generate DIST.## suitable for

ZPLOTTING size distributions at selected times; these size distributions may be average readings based on a selected interval of times. (I've been using 0.25 hour averages at each 0.5 hours from relative time of 0. into the run.)

*** SUMMARY of FILE TYPES ***

.E2R	raw eaa data from the pdp11	ASCII	volts	DO2EAA
.EA#	eaa currents file	ASCII	volts	SAVEAA
.EI#	eaa currents, t-interpolated	ASCII	volts	EAAINT
.ZE#	eaa data for ZPLOT	Binary	Number	XEAA or ZEAA
.ZI#	eaa data for ZPLOT, t-inter	Binary	Number	XEAA or ZEAA
.ZC#	eaa inverted currents	Binary	pAmps	
.ZN#	eaa inverted number densities	Binary	#/cc	
.ZV#	eaa inverted volume densities	Binary	um**3/cc	
.ES#	eaa inverted N,V,dpbar	ASCII		

If the identifying number, #, of the EAA file type is used, its meaning is as follows:

1 indicates EAA 132 sampling side A (or the whole bag),
2 indicates EAA 250 sampling side B (or the whole bag),
3 indicates EAA 132 sampling side B,
4 indicates EAA 250 sampling side A.

DO2EAA is an RT-11 program for taking dual EAA data.
SAVEAA is an RT-11 program for compressing raw EAA files.
VTEAA is an RT-11 program for converting EAA Volts to N display.
SHOEAA is an RT-11 program combining SAVEAA and VTEAA.
ZEAA is a VAX or PC program for GRAPHing EAA data stored in volts.
EAAINT is a VAX or PC program for time interpolation of EAA voltages.
THREATS is a PC program for inverting EAA data.
SELECT is a PC program for averaging and summarizing inverted EAA data.

After mid November, 1985, the new file types (.EA# & .EI#) using 11F7.4 datasets replaced the previous voltage files.

```
.ER,.E2R ==[SAVEAA]==>  .EA#                     EAA Raw Currents.

.EA#        ==[EAAINT]==>  .EI#                  EAA Interpolated Currents.

.EA#     ==[XEAA]==>    .ZE# ==[ZPLOT]==>  Raw N(K) v. Time.

.EI#     ==[XEAA]==>    .ZI# ==[ZPLOT]==>  Interpolated N(K) v. Time.

.EA#        ==[THREATS]==> .ZC#                  Inverted Current Profile
 or                        .ZN# ==[ZPLOT]==>  Inverted Chan N Profiles
.EI#                       .ZV# ==[ZPLOT]==>  Inverted Chan V Profile
                  DIST.n [ZPLOT]==>  Inv Raw Size Distributions
                  HIST.n     "           with Raw Histograms

.ZC#       ==[SELECT]==>   .ES# ==[ZPLOT]==>  Average Diameters v. Time
```

```
              .ZN# ==[ZPLOT]==>  Inverted Chan N Profiles
              .ZV# ==[ZPLOT]==>  Inverted Chan V Profile
          DIST.n [ZPLOT]==>  Inv Raw Size Distributions
          HIST.n    "          with Raw Histograms
```

*** AEROSOL DATA ACQUISITION AND PLOTTING FILES ***

TYPE	DESCRIPTION	FORMAT	UNITS	PARENT
.TOL	toluene data	ASCII	ppm	SAVTOL or EDT
.TOA, .TOB	ditto, sides A & B			
.CNC	cnc data	ASCII	Number	EDT
.CNA, .CNB	ditto, sides A & B			
.OPC	opc data, undiluted	ASCII	counts	SAVOPC
.OPA, .OPB	ditto, sides A & B			
.DOP	opc data, diluter on	ASCII	counts	SAVOPC
.DOA, .DOB	ditto, sides A & B			
.ZOP (.UOP)	opc data for ZPLOT	Binary	Number	XOPC
.ZOA, .ZOB	ditto, sides A & B			
.E2R	raw eaa data from the pdp11	ASCII	volts	DO2EAA
.EA# [.EV#]	eaa currents file	ASCII	volts	SAVEAA
.EI# [.IV#]	eaa currents, t-interpolated	ASCII	volts	EAAEAA
.EN# (.OE#)	eaa histogram file	ASCII	Number	VTEAA
.IN# (.IE#)	eaa histo file, t-interpolated	ASCII	Number	EAAINTRP
.ZE# (.E#)	eaa data for ZPLOT	Binary	Number	XEAA or ZEAA
.ZI# (.E#I)	eaa data for ZPLOT, t-inter	Binary	Number	XEAA or ZEAA
.ZC#	eaa inverted currents	Binary	pAmps	
.ZN#	eaa inverted number densities	Binary	#/cc	
.ZV#	eaa inverted volume densities	Binary	um**3/cc	
.ES#	eaa inverted N,V,dpbar	ASCII		

The identifying digit, #, of the EAA file type is as follows:
1 indicates EAA 132 sampling side A (or the whole bag),
2 indicates EAA 250 sampling side B (or the whole bag),
3 indicates EAA 132 sampling side B,
4 indicates EAA 250 sampling side A.

Programs:
EDT is a RT-11 and VAX editor for creating text files.
SAVTOL is an RT-11 program for entry of toluene peaks.
SAVOPC is an RT-11 program for entry of OPC counts.
DO2EAA is an RT-11 program for taking dual EAA data.
SAVEAA is an RT-11 program for compressing raw EAA files.
VTEAA is an RT-11 program for converting EAA Volts to N display.
SHOEAA is an RT-11 program combining SAVEAA and VTEAA.
XOPC is a VAX program for GRAPHing OPC count data.
XEAA is a VAX program for GRAPHing EAA data stored in volts.

ZEAA is a VAX program for GRAPHing EAA data stored in number.
EAAINTRP is a VAX program for time interpolation of raw EAA.
INTEAA is a VAX program for time interpolation of EAA voltages.

Notes:

In the interests of space efficiency and flexibility in processing the EAA data, I suggest use of the following:
SAVEAA instead of SHOEAA for compressing EAA data.
VTEAA instead of SHOEAA for video plotting.
EAAINT instead of EAAINTRP for time interpolation.
XEAA instead of ZEAA for making ZPLOTtable files.
Voltage files (.EA#,EI#) instead of Number files (.EN#,.IN#).

As of mid November, 1985, the new file types (.EA# & .EI#) using 11F7.4 datasets replaced the previous voltage files. All EAA programs are in the midst of being restructed so they obey this new standard, which is more compact, contains more useful identifying info, and can be sent easily across the network for use with IBM PC programs. Hence

```
.ER,.E2R ==[SAVEAA]==>  .EA#                 EAA Raw Currents.

.EA#        ==[EAAINT]==>  .EI#              EAA Interpolated Currents.

.EA#     ==[XEAA]==>    .ZE# ==[ZPLOT]==>  Raw N(K) v. Time.

.EI#     ==[XEAA]==>    .ZI# ==[ZPLOT]==>  Interpolated N(K) v. Time.

.EA#        ==[THREATS]==> .ET# ==[ZPLOT]==>  Inverted Dist v. Time.
                    DIST.n              Inv Raw Size Distributions
                    HIST.n              Raw Histograms

.EI#        ==[THREATS]==> .IT# ==[ZPLOT]==>  Inv & Int Dist v. Time.
                    DIST.n              Size Distributions.
                    HIST.n              Interpolated Histograms.
```

THREATS - THResholding Eaa Analysis with Twomey Smoothing

DOCUMENTATION FOR THE THREATS EAA INVERSION ROUTINE

THREATS -- THResholding Eaa Analysis with Twomey Smoothing is a modified Twomey routine for the inversion of EAA datasets. It incorporates the following features:

- an initial size distribution estimate using a simple histogram inversion of the measured EAA readings, obtained by neglecting channel cross-sensitivity.
- automatic adjustment of adjacent channels to compensate for negative channel readings, insuring that the target currents will not be negative, and additional adjustment to make target currents slightly positive.
- a smoothing routine which operates linearly in the current domain to take out unrealistic waviness which may result from the Twomey algorithm.
- a Twomey inversion routine assuming the full 40 element response matrix (sixteen sizes per power of ten in diameter), with automatic shutdown for each channel as the specified data error tolerances are met.
- iteration of the Smoothing-Twomey cycles until the curvature of the distribution (in the current domain) approaches a minimum, constrained by agreement with the measured data to within user selected tolerances.
- in addition to the above features, discussed by Greg Markowski (see the THREATS source comments for more information) and implemented in SMTWOM, the THREATS code optionally allows thresholding of the inverted current distribution against the noise level of the dataset, rapidly and smoothly reducing those parts of the size distribution which are actually lost in the noise. This option eliminates many ghost peaks which otherwise show up at the large size end of the volume distribution or the small size end of the number distribution, and which are identical to zero as far as the EAA's resolution is concerned.

The THREATS code was programmed in the second half of 1985 by Dale Warren (Ch.E., Caltech) for an IBM family PC in MICROSOFT's FORTRAN-77. It evolved from the SMTWOM code written by Greg Markowski for his Kaypro and cleaned up by DRW for the IBM PCs. The following files are relevant to THREATS, and contain further information about the program:

THREATS.FOR		FORTRAN Source File, with comments.
THREATS.EXE		Executable program, type "THREATS" to run.
*EAA.INP	Default	Input Data file. Enter your data here.
THREATS.INP		Documented EAA Raw Data file, usable as EAA.INP.
*EAA.TOL	Default	EAA Tolerances file, for your application.
THREATS.TOL		Documented EAA tolerances file, usable as EAA.TOL.
*EAA.OUT	Default	Text Output file, suitable for printing.
*DIST.1		Formatted EAA number and volume distributions.

*HIST.1	Formatted EAA histogram.
*PROFILE.ZN	Unformatted EAA number distribution with time.
*PROFILE.ZV	Unformatted EAA volume distribution with time.
EAA4.PDL	ZPLOT command file to generate plot foursome.
NUMA.PDL	ZPLOT command file to plot number in Chan 3-4.
NUMB.PDL	ZPLOT command file to plot number in Chan 5-7.
NUMC.PDL	ZPLOT command file to plot number in Chan 8-10.
NUMT.PDL	ZPLOT command file to plot total number in 3+,4+.
VOLA.PDL	ZPLOT command file to plot volume in Chan 3-4.
VOLB.PDL	ZPLOT command file to plot volume in Chan 5-7.
VOLC.PDL	ZPLOT command file to plot volume in Chan 8-10.
VOLT.PDL	ZPLOT command file to plot total volume in 3+,4+.
EAA.DOC	Summary of how EAA data is processed.
EPLOT.DOC	Summary of how EAA data is plotted.
SELECT.FOR	Program for further handling of inverted data.

Files denoted by an asterick (*) are user-created files. THREATS uses EAA.INP and EAA.TOL to generate EAA.OUT, DIST.1, and HIST.1 output files (if default names are used). The DIST and HIST files contain the calculated size distributions, in the following order of variables:

```
  DIST #1: Dp(i), diameter of ith size range [microns]
  DIST #2: dV/dlogDp(i), volume distribution by SMTWOM
  DIST #3: dN/dlogDp(i), number distribution by SMTWOM
  DIST #4: dVO/dlogDp(i), initial (or ideal) volume distribution
  DIST #5: dNO/dlogDp(i), initial (or ideal) number distribution
  DIST #6: dVh/dlogDp(i), volume distribution by Histogram
  DIST #7: dNh/dlogDp(i), number distribution by Histogram
  HIST #1: Dp(k), mean diameter of EAA channel k+1
  HIST #2: dVh/dlogDp(k), volume distribution by histogram
  HIST #3: dNh/dlogDp(k), number distribution by histogram
```

The user may type ZPLOT /MP:EAA4 to get a foursome of plots from DIST.1 and HIST.1. If more than one EAA dataset is entered, the nth data set will create DIST.n and HIST.n output files. Size distribution profiles of the entire file of EAA data may also be created, with names of PROFILE.ZN and PROFILE.ZV. These are unformatted files suitable for ZPLOTting (see NUM*.PDL and VOL*.PDL), consisting of the Time, Histogram 2-10, and the Inverted Distribution subchannels 1-37, plus totals by both inversion methods.

By default, THREATS goes through 10 loops, each consisting of up to 30 Twomey iterations (stopping on each channel as soon as the error tolerance is met) and 1 sub-current distribution smoothing (1/2 prior value + 1/2 neighbors' average value). Alternately (using Markowski's suggestions), the routine can be made to stop when the prior two iterations have decreased the curvature by less than 2.5%. The final exit occurs after a set of Twomey iterations (not smoothings, as then error tolerance might not be met). The user may override these conditions if he really wants to do so, as described in THREATS.TOL.
By default, no Thresholding is done. To use thresholding,

set IGNORE to 1 in your .TOL file and select a noise cutoff for subchannel current; 0.0004 is about as high as I would recommend (that means an equivalent channel current of 0.016 is the largest current that will get treated as noise); larger values might work, but they can bias the distribution and ignore true signals. It may be worthwhile to invert your EAA datasets with and without thresholding to make sure that the dominant modes of the distribution are not altered in size. The appropriate thresholding and tolerances will depend upon the stability of your aerosol and the precision with which the EAA data was taken.

If THREATS is working properly, the error tolerance will usually be met after a reasonable number of Twomey iterations, and successive loops will converge on a stable result fairly quickly. The resulting distribution will normally be noticeably sharper (not to mention smoother) than the Histogram (first approximation) method for EAA inversion discussed in the EAA manual. If the program has to keep using the maximum number of Twomey iterations (normally 30), then the data are not meeting the tolerances which you have assumed, and the inverted distributions are not likely to be very smooth.

THREATS also allows the user to generate composite log-normal distributions for testing of the code with "perfect" EAA data.

-- v3.7, 1/16/86 Dale R. Warren

EAA PLOTTING using ZPLOT

There are two frequently useful ways of plotting EAA data. One is the size distribution against particle diameter; either number or volume, by histogram or (semi-)continuous inversion. The second is the time profile; again, either number or volume, by histogram or more sophisticated inversion method. The size distribution is more common, but, it is only a snapshot in time of an evolving aerosol distribution, so a time profile of concentrations of one of more sizes of particles against time may more concisely represent aerosol size evolution.

When size distributions are of interest, our plottable EAA size distributions are stored in DIST.# and HIST.# formatted data files, where "#" is an integer identifying the dataset. The volume distribution is plotted as log (d V / d log10(dp)) with V in units of um**3/cm**3, while the number distribution is plotted as log (d N / d log10(dp)) with N in units of number per cc.
The inverted DIST.# files have the following structure:

```
  1         2      3      4      5      6      7
Dp(Chav)   V      N      Vs     Ns     Vh     Vn
```

If channels 2-10 are inverted, the file will be 37 records long.
("s" refers to starting and "h" refers to histogram.)

The histogram HIST.# files have the following structure:

```
  1          2          3
Dp(Chmin)   Vol        Num
Dp(Chmax)    "          "
```

The above pattern is repeated for each of the ten channels, making the HIST.# file 20 records long.

When time profiles are of interest, the plottable EAA size distributions are stored in *.ZN and *.ZV unformatted data files, holding particle number and particle volume, respectively.
The *.ZN files have the following structure:

```
1      2-10                11-47                  48-50      51-53
Time   N(K) by Histogram   N(I) by Inversion     HISNUM    TOTNUM(2+,,4+)
```

The *.ZV volume files have a parallel structure, as follows:

```
1      2-10                11-47                  48         49
Time   V(K) by Histogram   V(I) by Inversion     HISVOL     TOTVOL
```

To save space, the default storage file is only the *.ZC current difference files, which have the following parallel structure:

```
1      2-10                11-47
Time   C(K) by EAA exp.    C(I) by Inversion     (No totals)
```

Note that SELECT operates on the *.ZC files, and can regenerate the *.ZN and *.ZV files quickly, as needed.

Comparable columns for ZN, ZV, and ZC files are as follows:

Histogram (Channel)	2	3	4	5	6	7	8	9	10	T=48
Inversion (col)	13	17	21	25	29	33	37	41	45	T=51

One final type of data storage file is used, the EAA Summary file of type *.ES (default SEE.ES). It has the following formatted columns of values:

1	2	3	4	5	6	7	8	9	10
Time	Number	Volume	Dp	Dp	Dp	Dp	Dp	SG	AGREE
hrs	Total	Total	Num	Num	Vol	Vol	Vol	log-	
into	invert	invert	mean	mode	mode	mean	log-	mean	

The currently available ZPLOT command files (PDL files) are as follows:

N1.PDL - Plots a full page number distribution, both as inverted by THREATS and by the simple histogram. Data files are DIST.1 and HIST.1

V1.PDL - Plots a full page volume distribution, both as inverted by THREATS and by the simple histogram. Data files are DIST.1 and HIST.1

EAA4.PDL - Plots the inverted and histogram size distributions on four diagrams at once: dN/dDp, dN/dlogDp, dV/dDp, dV/dlogDp. Data files are DIST.1 and HIST.1

ET4.PDL - Same foursome as EAA4.PDL, except this is designed for test cases, where the "true" distribution is known, or where the THREATS initial guess is of interest; this third size distribution is also plotted. Data files are HIST.1 and DIST.1

VOLT.PDL - Plots the total volume, inverted and by histogram method, as a function of time. Data file is PROFILE.ZV.

NUMT.PDL - Plots the total number, by THREATS inverstion and by simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZN.

NUMA.PDL - Plots the number density functions of EAA Channels 3 and 4, by THREATS inversion and by the simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZN.

NUMB.PDL - Plots the number density functions of EAA Channels

5, 6, and 7, by THREATS inversion and by the simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZN.

NUMC.PDL - Plots the number density functions of EAA Channels 8, 9, and 10, by THREATS inversion and by the simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZN.

VOLA.PDL - Plots the volume density functions of EAA Channels 3 and 4, by THREATS inversion and by the simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZV.

VOLB.PDL - Plots the volume density functions of EAA Channels 5, 6, and 7, by THREATS inversion and by the simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZV.

VOLC.PDL - Plots the volume density functions of EAA Channels 8, 9, and 10, by THREATS inversion and by the simple HISTOGRAM approach, as a function of time. The data file is PROFILE.ZV.

N9.PDL - Plots a time series of nine inverted number distributions, normally generated by SELECT. The data files are DIST.1 through DIST.9.

V9.PDL - Plots a time series of nine inverted volume distributions, normally generated by SELECT. The data files are DIST.1 through DIST.9.

DP.PDL - Plots the characteristic diameters of the inverted EAA size distribution as a function of time. The characteristic diameters are defined as (in order of appearance) Number-mean, Number-mode, Volume-mode, Volume-mean, and Volume-log-mean, in microns, over the EAA operational range of 0.01 to 1.00 microns. The data file is of the *.ES EAA summary type and must be specified by the user in ZPLOT.

LOGFIT.PDL - Plots the logmean diameter with the Number-mode and Volume-mode on the upper plot, against the calculated Geometric Standard Deviation and the fractional overlap (0. to 1.) between the logmean and the THREATS distributions on the lower plot. (E.g., a geometric standard deviation of 1.2 means 68% of the total volume lies at diameters between 0.833 and 1.20 of the logmean diameter; Davies considers any distribution with SG<1.2 as practically monodisperse!) The data file is of the *.ES type, and must be specified by the user in ZPLOT.

RAW.PDL - Plots the raw currents (not differences) of all EAA channels as a function of time. The data file is of the .EA# type, except the two header lines have been stripped. (The second line of the *.EA# file must show the channel range was from 2 to 11, and not be differenced, so the line begins 2,11,0)

The learning feature has been used in ZPLOT to make it easier for the user to generate appealing plots of exactly the range of interest. Use <CTRL>G to Go to one of the ten learnt MACRO-like routines, which are currently programmed as follows:

2 - Set X range to from 0.01 to 1.00 (microns) and turn diagram frame on.
4 - Set ranges automatically of the foursome of plots called by EAA4.PDL and others. Also turns frame on.
5 - Change X range from 0.0 to 5.0 (hours) exactly, turning off neating, 5 x-divs, diagram frame on.

```
      PROGRAM ZEAA
C
C      VAX FORTRAN-77 or MICROSOFT FORTRAN-77 PROGRAM
C      Simple Histogram Inversion of EAA Data.
C      Transfers EAA Current Datasets to ZPLOT Histogram Evolution File
C      File Types:    .EA# or .EI#     .ZE# or .ZI#
C
      CHARACTER*20 IFILE,OFILE
      COMMON /EAA/ EAADP(10),EAADPM(9)
      REAL V(10), PN(9), PV(9)
C
C      V(K) = Volts   PN(K) = Particle Number    PV = Particle Volume
C       current, pA              #/cc                      um**3/cc
C
C      Remember:  1 cu.um./cc = 1.E-12 = 1 ug/cu.m. at 1 g/cc density
C
C      Element K refers to EAA Channel K+1 for all arrays.
C
      WRITE(*,10)
 10   FORMAT(/' Enter Input EAA voltage File Name: '\)
      READ(*,15) IFILE
 15   FORMAT(A20)
      OPEN (11,FILE=IFILE,STATUS='OLD')
      WRITE(*,20)
 20   FORMAT(/' Enter ZPLOTtable Output Filename (.ZE#) : '\)
      READ(*,15) OFILE
      OPEN(12,FILE=OFILE,STATUS='NEW')
C
C      SET UP EAA COMMON VALUES
C
      CALL SETEAA
C
      N=0
C
100   READ(11,80,END=500,ERR=400) HOURS,(V(I),I=1,10)
 80   FORMAT(F8.4,2X,10(F7.4,1X))
      N=N+1
C     WRITE(*,80) HOURS,(V(I),I=1,10)
      CALL EAASIZ(V,PN,PV,TN,TV,TS,TD)
C
C      Voltages -> Number and Volume
C
      TN3=TN-PN(1)
      TN4=TN3-PN(2)
      WRITE(12) HOURS,(PN(I),I=1,9),TN,TN3,TN4,TV
      GOTO 100
C
400   STOP 'STOPPING ON READ ERROR'
500   CLOSE(11)
      CLOSE(12)
      WRITE(*,*) N,' EAA DATA SETS TRANSFERRED'
      STOP 'NORMAL COMPLETION'
```

```
        END

      SUBROUTINE SETEAA
C
C       Call Initially to Setup for EAA data analysis:
C
C       Sets EAA Fixed Channel Parameters
C       Note Subscript I refers to channel I+1, as channel 1 not used
C       Thus there are ten voltage readings per EAA dataset.
C
      COMMON /EAA/ EAADP(10),EAADPM(9),EAACON(9),EAASN(9),EAAVN(9)
      DATA PI / 3.141593 /
C
C       Set Channel Boundaries in Diameter, microns
C
      DO 10 I=1,10
   10   EAADP(I)=10.**(0.25*FLOAT(I-10))
C
C       Set Channel Mean Diameters, microns
C
      DO 20 I=1,9
   20   EAADPM(I)=SQRT(EAADP(I)*EAADP(I+1))
C
C       Set Surface to Number (um**2) and Volume to Number (um*3) Ratios
C
      DO 30 I=1,9
        EAASN(I)=PI*EAADPM(I)**2
30      EAAVN(I)=PI/6.*EAADPM(I)**3
C
C       Set Diagonal Response Matrix, #/cc per picoamp (volt)
C       This is the simple HISTOGRAM inversion method.
C
      EAACON(1)=9.52E6
      EAACON(2)=4.17E5
      EAACON(3)=1.67E5
      EAACON(4)=8.70E4
      EAACON(5)=4.44E4
      EAACON(6)=2.41E4
      EAACON(7)=1.23E4
      EAACON(8)=6.67E3
      EAACON(9)=3.51E3
      RETURN
      END
C
C
      SUBROUTINE EAASIZ(V,PN,PV,TN,TV,TS,TD)
      COMMON /EAA/ EAADP(10),EAADPM(9),EAACON(9),EAASN(9),EAAVN(9)
C
C       Input EAA Voltage Set, returns aerosol size distribution
C
      REAL V(10),PN(9),PV(9),DELV
C
C       Initialize summations
```

```
C
      TN=0.
      TV=0.
      TS=0.
      TD=0.
C
C
C       For EAA Channels 2 through 10:
C         DELV = current difference (zero if negative)
C         PN = Particle Number, #/cc
C         PV = Particle Volume, um**3/cc [ug/m**3 if unit density]
C         TN = Total Number      TV = Total Volume
C         TS = Total Surface     TD = Total Length
C
      DO 10 I=1,9
        DELV=V(I)-V(I+1)
        IF (DELV.LT.0.) DELV=0.
        PN(I)=EAACON(I)*DELV
        PV(I)=EAAVN(I)*PN(I)
        TN=TN+PN(I)
        TV=TV+PV(I)
        TS=TS+PN(I)*EAASN(I)
        TD=TD+PN(I)*EAADPM(I)
10    CONTINUE
      RETURN
      END
      PROGRAM EAAINT
C
C  This program interpolates the EAA currents back to a
C   common time for each successive pair of readings.
C  This cancels out the generally small error in current
C   differences due to systematic drift of the voltages
C   with time (due to wall losses, growth, etc.).
C  (Nonadjacent pairs of datasets are not interpolated.)
C
C  Files are formatted by the new .EA# and .EI# standards,
C   where the input .EA# file must contain raw currents and
C   the output .EI# file will usually be current differences.
C   Units are picoamps (measured as volts).
C
C   LINE 1>  Text                          A70
C   LINE 2>  ICHAN,LCHAN,KDIF,MAXSET       Free Format
C   LINES+>  HOUR,CSET(K),K=1,KMAX         11F7.4
C
C        -- DRW in NOV-85 using MICROSOFT FORTRAN-77 for IBM-AT
C
      CHARACTER*20 RNAME,INAME
      CHARACTER*60 TEXT
      CHARACTER*1 ASK
      REAL C(10),C1(10),C2(10),TSET(10),TOFF(10)
      DATA TSET / 30.,20.,16.,10.,6.,6.,6.,4.,4.,4. /
      DATA TREAD / 5.3 /
      DATA TLAG / 5.1 /
C
C  Default timing was used in the Summer of 85 by DO2EAA.
C  Cycle time worked out to 164.1 seconds.
C  TSET contains the computer-controlled settling times.
C
C  Subscript K refers to EAA Channel K+1.
C  The reference time is the end of the earlier cycle.
C
      TCYCLE = TLAG
      DO 100 K=1,10
         TCYCLE = TCYCLE + TSET(K) + TREAD
 100     TOFF(K) = TCYCLE - 0.5*TREAD
C
C  Normalize TOFF as fractional time into cycle.
C
      DO 150 I=1,10
 150     TOFF(I) = TOFF(I) / TCYCLE
C
C  Set NDIF = 1 to save interpolated data as current differences.
C  Else NDIF = 0 will save as interpolated currents.
C
      WRITE(*,155)
 155  FORMAT(/' Save as EAA channel current differences? [Y] '\)
      READ(*,156) ASK
 156  FORMAT(A1)
```

```
      NDIF = 1
      IF (ASK.EQ.'N' .OR. ASK.EQ.'n') NDIF = 0
      IF (NDIF.EQ.0) THEN
         KMAX = 10
      ELSE
         KMAX = 9
      ENDIF
C
      WRITE(*,160)
 160  FORMAT(/' EAA Currents (.EA#) Input Filename: '\)
      READ(*,165) RNAME
 165  FORMAT(A)
      WRITE(*,170)
 170  FORMAT(/' EAA Interpolated Currents (.EI#) Output Filename: '\)
      READ(*,165) INAME
      IKOUNT = 1
      KOUNT = 0
      OPEN(2,FILE=RNAME,STATUS='OLD',FORM='FORMATTED')
      OPEN(3,FILE=INAME,STATUS='NEW',FORM='FORMATTED')
      READ(2,222) TEXT
 222  FORMAT(A60)
      WRITE(*,*) ' '
      WRITE(*,*) TEXT
      WRITE(3,223) TEXT
 223  FORMAT(' Interpolated ',A)
      READ(2,*) ICHAN,LCHAN,KDIF,MAXSET
      WRITE(*,230) ICHAN,LCHAN,KDIF,MAXSET
 230  FORMAT(/' ICHAN =',I2,3X,'LCHAN =',I3,3X,'KDIF =',I2,3X,
     #        'MAXSET =',I4/)
      IF (KDIF.NE.0) STOP 'NEED RAW CURRENTS TO INTERPOLATE'
      IF (NDIF.EQ.0) THEN
         NCHAN=LCHAN
      ELSE
         NCHAN=LCHAN-1
      ENDIF
      WRITE(3,*) ICHAN,NCHAN,NDIF,0
      READ(2,301) T1,(C1(K),K=1,10)
C
C  Read in Current Dataset from File
C
 300  READ(2,301,END=900) T2,(C2(K),K=1,10)
 301  FORMAT(12F7.4)
      IKOUNT = IKOUNT + 1
C
C  Skip if not actually next cycle.  TCY may be off by 0.4 sec.
C
      TCY = 3600.* (T2 - T1)
      IF (ABS(TCY-TCYCLE)/TCYCLE .GT. .02) GOTO 550
      KOUNT = KOUNT + 1
C
      DO 500 K=1,10
 500     C(K) = C2(K) + TOFF(K) * ( C1(K) - C2(K) )
C
```

```
C  Write the interpolated dataset to disk.
C  Save as current differences if NDIF<>0
C  The time is for the end of the first cycle, T1
C
      IF (NDIF.NE.0) THEN
        DO 505 K=1,KMAX
 505        C(K) = C(K) - C(K+1)
      ENDIF
      WRITE(3,501) T1,(C(K),K=1,KMAX)
 501  FORMAT(12F7.4)
C
C  Onward, as the new becomes old . . .
C
 550  DO 600 K=1,10
 600     C1(K) = C2(K)
      T1 = T2
      GOTO 300
C
 900  CLOSE (2)
      CLOSE (3)
      WRITE(*,950) IKOUNT,KOUNT
 950  FORMAT(/' EAA Datasets Transferred:  IN =',I4,'   OUT =',I4 /)
      STOP 'EAAINT DONE'
      END
      PROGRAM THREATS
C
C  MARKOWSKI-TWOMEY EAA INVERSION ALGORITHM
C
C  THIS PROGRAM INVERTS ELECTRICAL AEROSOL ANALYZER (EAA) DATA.
C  Either EAA current readings or the parameters of a hypothetical
C    volume distribution may be input by the user.
C  The initial guess is based on the target currents, subdivided
C    and smoothed, assuming no cross-sensitivity.
C  The Twomey algorithm is used to correct the initial guess
C    until the trial sub-currents yield a response within the
C    chosen tolerance of the target currents.
C  The distribution then may go through a threshold discriminator
C   which pulls sections of distribution whose signal is below
C   the noice level down to insignificant values, thus eliminating
C   small ghost peaks, generally at the tails.  This is optional.
C  The distribution is then numerically smoothed, and the
C    Twomey and smoothing steps are repeated until the
C    process begins to converge.
C  The next EAA current data set is then processed in the same way.
C
C  Based on Greg Markowski's 1985 program EAATW1 for his KAYPRO.
C  Extensively reprogrammed in Fall 1985 by Dale Warren (Caltech)
C    for an IBM family PC using MICROSOFT FORTRAN-77.
C  Error tolerance selection semi-automated.
C  Input data defaults to EAA.TOL and EAA.INP files (see those
C    example input files for usage information).
C  Use SELECT to manipulate the .ZC output files for ZPLOTting
C    of channel profiles and distributions at selected times.
C  See THREATS.DOC for more information.
C
C
C  REFERENCES:
C  1.  S. TWOMEY, "COMPARISON OF CONSTRAINED LINEAR INVERSION
C        AND AN ITERATIVE NONLINEAR ALGORITHM APPLIED TO THE
C        INDIRECT ESTIMATION OF PARTICLE SIZE DISTRIBUTIONS,"
C        J. COMP. PHYS., 18:188-200 (1979).
C  2.  GREGORY R. MARKOWSKI, "IMPROVING TWOMEY'S ALGORITHM
C        FOR INVERSION OF AEROSOL MEASUREMENT DATA,"
C        AEROSOL SCI & TECHNO, in press (1985).
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /CHAN/ KDIV,MAG
      COMMON /FLAG/ IPRNT,INFO,INFOT
      COMMON /TEST/ DG(3),SG(3),VM(3)
      COMMON /INFO/ HOUR,MAXSET,KFIRST,KLAST,OFILE,HEAD(2)
      COMMON /SIZE/ DIA(41),DEAA(10)
      COMMON /SET/ TARGET(10),CTOL(10),RFIT(10),AFIT(10)
      COMMON /INVRT/ MAXTWO,MAXSMO,ISMAX,MINSMO,KDIF,SMTOL
      COMMON /DROP/ NDZERO,IGNORE,CMIN,TNOISE,DROP
      COMMON /STATS/ JSET,JLOOP,JTWO,JCON,SUMSIG,BEGSIG,SUMCUR,BEGCUR
      CHARACTER*20 OFILE,DFILE,SFILE,NFILE,VFILE,CFILE
      CHARACTER*70 HEAD
      CHARACTER*4 DNAME,HNAME
      LOGICAL INFO,INFOT,SAVDIS,SAVPRO,SAVCUR
      CHARACTER*1 ASK
      DIMENSION COUT(10), HISTV(10), HISTN(10)
      DIMENSION TRIAL(41), ATRIAL(41)
      REAL SDIST(41), TDIST(41), RAT(41), NDIST(41)
      DIMENSION TOTNUM(3),STNUM(3),HISNUM(3)
      DIMENSION RM(10,41), RI(41,10), RA(41,10)
      DIMENSION RMX(240), RMY(160)
      EQUIVALENCE (RMX(1),RM(1,1)) , (RMY(1),RM(1,25))
C
C  RM was split into RMX and RMY so DATA had <10 Continuations
C
      DATA RMX / 0.5,9*0., 1.,9*0., 1.,9*0., 1.,9*0., 0.5,0.5,8*0.,
     1   0.,1.,8*0., 0.,1.,8*0., 0.,1.,8*0., 0.,.48,.52,7*0.,
     2   0.,.05,.95,7*0., 0.,.08,.92,7*0., 2*0.,1.0,7*0.,
     3   2*0.,.51,.48,6*0., 2*0.,.44,.52,.04,5*0.,
     4   2*0.,.49,.07,.44,5*0., 2*0.,.15,.48,.19,.17,4*0.,
     5   2*0.,.14,.35,.32,.17,4*0., 2*0.,.03,.34,.51,.08,.02,3*0.,
     6   3*0.,.14,.65,.17,.02,.02,2*0., 3*0.,.05,.61,.31,.03,3*0.,
     7   4*0.,.51,.40,.08,3*0., 4*0.,.45,.48,.07,3*0.,
     8   4*0.,.40,.51,.09,3*0., 4*0.,.13,.67,.17,.03,2*0. /
      DATA RMY /
     1   4*0.,.03,.47,.38,.08,.04,0., 5*0.,.35,.46,.14,.04,.01,
     2   5*0.,.22,.48,.20,.06,.04, 5*0.,.15,.45,.25,.09,.06,
     3   5*0.,.08,.42,.32,.11,.07, 5*0.,.02,.32,.38,.18,.10,
     4   6*0.,.31,.36,.18,.15, 6*0.,.20,.39,.21,.20,
     5   6*0.,.19,.30,.26,.24, 6*0.,.13,.27,.25,.35,
     6   6*0.,.11,.28,.25,.36, 6*0.,.07,.23,.25,.45,
     7   6*0.,.04,.20,.26,.50, 6*0.,.02,.18,.25,.55,
     8   7*0.,.15,.26,.59, 7*0.,.13,.25,.62/
C
C   NOTE THAT THE RESPONSE MATRIX RM CONTAINS 40 SIZES BY 10 CHANNELS
C   THESE SIZES RANGE FROM .0056 TO 1.54 MICRONS OR EAA 2.0 TO 10.75
C   AS LONG AS RM IS FIXED, SUBSCRIPT N=1 REFERS TO START OF CHANNEL 2
C   SIMILARLY, IT IS ASSUMED THAT SUBSCRIPT K=1 REFERS TO EAA CHANNEL 2
C   AND MAG=4 (4 DIVISIONS PER EAA CHANNEL) WILL BE USED.
C   THE USER MAY SELECT WHAT SUBSET OF THESE 10 CHANNELS ARE IN USE.
C
C  Version 1.0 released 31-OCT-85 ==> SMTWOM
C        Version 2.0 finished  4-NOV-85
C  Version 3.0 finished 14-NOV-85 ==> THREATS
C  Version 3.7 is dated 15-JAN-86          "
C
      VERSON = 3.7
      MXDIV = 41
      MAG = 4
      KDIV = 4 * MAG
      EAAHLF = 10.**0.125
C
      WRITE(*,60) VERSON
   60 FORMAT(/20X,'THREATS v',F3.1,' EAA INVERSION'/)
```

```
C
C         SET DIAMETERS      Beginning with Channel 2 of EAA
C
      DIA(1) = 0.005623413
      DO 100 I = 2,41
 100     DIA(I) = DIA(I-1) * 1.154782
      DEAA(1) = 0.007498942
      DO 110 K = 2,10
 110     DEAA(K) = DEAA(K-1) * 1.778279
C
C         TRANSFER RM TO RI AND TO RA IN PROPER ORDER
C
      DO 120 I = 1,41
         DO 115 K = 1, 10
              RA(I,K) = RM(K,I)
              RI(I,K) = RM(K,I)
 115     CONTINUE
 120  CONTINUE
C
C         ADJUST ELEMENTS IN RA FOR SMOOTHING TO HELP TWOMEY INVERSION
C
      RA(15,3) = .40
      RA(15,4) = .28
      RA(15,5) = .32
C
C
C          WHAT INFO SHOULD BE SAVED FROM THE INVERSION?
C
C  Current defaults will save the Current Profile (.ZC)
C  on which the SELECT program may operate, to create the
C  Number (.ZN) or Volume (.ZV) profiles, time-averaged
C  distributions (DIST.#), or size summary files (.ES).
C
      WRITE(*,600)
 600  FORMAT(/' SAVE N & V PROFILES [N] ? '\)
      READ(*,666) ASK
      SAVPRO=.FALSE.
      IF (ASK.EQ.'Y' .OR. ASK.EQ.'y') SAVPRO=.TRUE.
      WRITE(*,601)
 601  FORMAT(' SAVE CURRENT PROFILE [Y] ? '\)
      READ(*,666) ASK
      SAVCUR=.TRUE.
      IF (ASK.EQ.'N' .OR. ASK.EQ.'n') SAVCUR=.FALSE.
      WRITE(*,602)
 602  FORMAT(' SAVE EACH DISTRIBUTION AND HISTOGRAPH [N] ? '\)
      READ(*,666) ASK
      SAVDIS=.FALSE.
      IF (ASK.EQ.'Y' .OR. ASK.EQ.'y') SAVDIS=.TRUE.
      WRITE(*,604)
 604  FORMAT(' PRINT TWOMEY STATS [Y] ? '\)
      READ(*,666) ASK
      INFOT=.TRUE.
      IF (ASK.EQ.'N' .OR. ASK.EQ.'n') INFOT=.FALSE.
      WRITE(*,606)
 606  FORMAT(' PRINT GENERAL INFO [N] ? '\)
      READ(*,666) ASK
      INFO=.FALSE.
      IF (ASK.EQ.'Y' .OR. ASK.EQ.'y') INFO=.TRUE.
      WRITE(*,608)
 608  FORMAT(' PRINT INTERMEDIATE DIST (Y/N/S/T) [N] ? '\)
      READ(*,666) ASK
      IPRNT=0
      IF (ASK.EQ.'Y') IPRNT=10
      IF (ASK.EQ.'S') IPRNT=2
      IF (ASK.EQ.'T') IPRNT=1
 666  FORMAT(A1)
      WRITE(*,*) ' '
C
      DNAME='DIST'
      HNAME='HIST'
      NFILE='PROFILE.ZN'
      VFILE='PROFILE.ZV'
      IF (SAVPRO) THEN
        OPEN (11,FILE=NFILE,STATUS='NEW',FORM='UNFORMATTED')
        OPEN (12,FILE=VFILE,STATUS='NEW',FORM='UNFORMATTED')
      ENDIF
      CFILE='CURRENT.ZC'
      IF (SAVCUR) THEN
        OPEN (20,FILE=CFILE,STATUS='NEW',FORM='UNFORMATTED')
      ENDIF
C
C          GET INPUT PARAMETERS by INPUT SUBROUTINE
C
      NOWSET = 0
 200  CALL INPUT (NOWSET, IERR)
      IF (IERR.GT.0) THEN
        WRITE(*,*) 'ERROR IN INPUT ROUTINE'
   GOTO 990
      ELSE IF (IERR.LT.0) THEN
        WRITE(*,*) 'END OF EAA DATA'
        GOTO 990
      END IF
      IF (NOWSET.EQ.1) WRITE(6,60) VERSON
C
C          USE FOLLOWING IF TARGET WAS NOT ENTERED AS AS A DATA SET;
C     ON KDIF<0, GENERATE SIZE DISTRIBUTION AND CALCULATE
C     TARGET FROM DISTRIBUTION PARAMETERS AND RESPONSE MATRIX.
C           INIT sets SDIST array to zero.
C           LOGNRM adds lognormal modes to SDIST (volume).
C           GETSIZ converts SDIST volume to pseudocurrent TRIAL.
C           RESP finds response TARGET from pseudocurrent TRIAL.
C
      IF ( KDIF .LT. 0 ) THEN
        WRITE(6,133) ( I,DG(I),SG(I),VM(I), I=1,3 )
 133    FORMAT(' MODE ',I2, ':   DG =',F6.3,'   SG =', F5.2,
     #          '   Vm =',F11.6)
```

```
        CALL INIT ( SDIST, NMAX, 0. )
        CALL LOGNRM ( 1, SDIST )
        CALL LOGNRM ( 2, SDIST )
        CALL LOGNRM ( 3, SDIST )
        CALL GETSIG ( SDIST, ATRIAL )
        CALL RESP ( ATRIAL, TARGET, RI )
        IF (INFO) THEN
          WRITE(6,*) ' CALCULATED SUM OF LOGNORMAL DISTRIBUTIONS'
          WRITE(6,624)
          WRITE(6,625) (I, DIA(I), SDIST(I), ATRIAL(I), I=NMIN,NMAX)
        END IF
 624    FORMAT ( 2('   I      Dp  dV/dlDp ATRIAL ',2X))
 625    FORMAT ( 2(I4,F8.4,F10.4,F8.4,2X) )
      END IF
C
C  WRITE OUT CURRENTS FOR EACH CHANNEL IN TABLE
C  CALCULATE CURRENT TOLERANCES FROM RFIT(K) AND AFIT(K)
C
      WRITE(*,112) (K+1, K=KFIRST,KLAST)
      WRITE(6,112) (K+1, K=KFIRST,KLAST)
      IF (NOWSET.LE.1) THEN
         WRITE(6,117) (RFIT(K),K=KFIRST,KLAST)
         WRITE(*,117) (RFIT(K),K=KFIRST,KLAST)
         WRITE(6,118) (AFIT(K),K=KFIRST,KLAST)
         WRITE(*,118) (AFIT(K),K=KFIRST,KLAST)
      END IF
      WRITE(*,210) (TARGET(K), K=KFIRST,KLAST)
      WRITE(6,210) (TARGET(K), K=KFIRST,KLAST)
C
C  Setting minimums can be critical to tails of distribution.
C  CMIN is minimum initial value for a channel current (if <=0)
C  TNOISE is the minimum trial subcurrent which is believed
C   to possibly represent signal.  If a trial current goes
C   below TNOISE and NDZERO is operational, THREATS will attempt
C   to send the TRIAL value towards zero using a factor of
C   DROP per subcurrent inverval.  The smoothing and
C   Twomey routines may then work on the curve.
C
      IF (KDIF.EQ.0) THEN
         DO 130 K=KMIN,KMAX
 130        TARGET(K)=TARGET(K)-TARGET(K+1)
         WRITE(*,210) (TARGET(K),K=KMIN,KMAX)
         WRITE(6,210) (TARGET(K),K=KMIN,KMAX)
      END IF
 112  FORMAT (/'  CHANNEL',10I7 )
 117  FORMAT (' RFIT TOL', 10F7.3 )
 118  FORMAT (' AFIT TOL', 10F7.3 )
 210  FORMAT (' CURRENT ',10F7.3)
 211  FORMAT (' Current ',10F7.3)
C
C  Get rid of negative values, then set minimum value.
C
      CALL NONEG(TARGET)
      DO 140 K = KMIN,KMAX
         IF (TARGET(K).LE.CMIN) TARGET(K)=CMIN
 140     CTOL(K) = RFIT(K) * TARGET(K) + AFIT(K)
      WRITE(6,211) (TARGET(K),K=KMIN,KMAX)
      WRITE(*,211) (TARGET(K),K=KMIN,KMAX)
      WRITE(*,104) (CTOL(K),K=KMIN,KMAX)
      WRITE(6,104) (CTOL(K),K=KMIN,KMAX)
 104  FORMAT(' CTOL',4X,10F7.3)
      WRITE(6,*) ' '
      WRITE(*,*) ' '
      IF (INFO) WRITE (6,105) KMIN+1,KMAX+1,KDIV,NMIN,NMAX
 105  FORMAT(/' EAA Channels ',I2,'-',I2,' using KDIV=',I3,
     #             ' for Sub-Divisions ',I2,'-',I2 /)
C
C        CALCULATE BEGINNING GUESS SUB-CURRENTS
C
      DO 150 I = NMIN,NMAX
         K = (I-1)/MAG +1
         TRIAL(I) = TARGET(K)/FLOAT(MAG)
 150  CONTINUE
C
C        SMOOTH INITIAL GUESS
C
      BX = TRIAL(NMAX-1)
      IF (NMAX .LT. 41) TRIAL(NMAX)= BX*BX/TRIAL(NMAX-5)
      CALL SMOOTH ( TRIAL, NMIN, NMAX, NDZERO )
      IF (INFO) WRITE (6,*) ' BEGINNING TRIAL SUB-CURRENTS'
      IF (INFO) WRITE (6,640) (I, TRIAL(I), I = NMIN,NMAX)
 640  FORMAT ( 4(I4,F8.3) )
C
C        WRITE IDEAL RESPONSE MATRIX
C
      IF (INFO.AND.NOWSET.EQ.1) WRITE (6, 610 )
 610  FORMAT ( // 26X, ' RESPONSE MATRIX ')
      IF (INFO.AND.NOWSET.EQ.1) WRITE (6,611) (I, I = 2,11 )
 611  FORMAT (11X, 10I6)
      IF (INFO.AND.NOWSET.EQ.1) WRITE(6,613) (I,DIA(I),
     #                           (RA(I,K), K=1,10 ), I=NMIN,NMAX)
 613  FORMAT (I3,F7.4,2X,10F6.2)
C
C        SAVE FIRST TRIAL CURRENTS
C  FIND & PRINT SIZE DISTRIBUTION (OVER KDIV)
C  NOTE SDIST IS STARTING TRUE DISTRIBUTION IF KDIF<0
C
      IF (KDIF.GE.0) THEN
        DO 190 I = NMIN,NMAX
 190       ATRIAL(I) = TRIAL(I)
        CALL GETVOL ( ATRIAL, SDIST, STVOL, STNUM )
      ELSE
        CALL GETVOL ( ATRIAL, RAT,   STVOL, STNUM )
      END IF
C
      CALL HISTO ( TARGET, HISTV, HISTN, HISVOL, HISNUM )
      IF (INFO) WRITE (6,*) ' TRIAL (STARTING SUBCURRENTS) :'
      IF (INFO) WRITE (6,618) (TRIAL(I), I=NMIN,NMAX)
      IF (INFO) WRITE (6,*) ' SDIST (STARTING VOLUME DISTRIBUTION) :'
      IF (INFO) WRITE( 6,618) (SDIST(I), I=NMIN,NMAX)
      IF (INFO) WRITE(6,*) ' TARGET (TRUE EAA CURRENTS) :'
      IF (INFO) WRITE (6,615) (TARGET(K),K=KMIN,KMAX)
 615  FORMAT (10F7.3)
 618  FORMAT (8F8.3)
C
C           PRELIMINARIES DONE . . . INVERT THE EAA DATA!
C
C  INVERT REPEATEDLY APPLIES THE TWOMEY ALGORITHM AND SMOOTHS
C
      CALL INVERT (TRIAL, COUT, RI, RA)
C
C  GET CALCULATED VOLUME DISTRIB IN TDIST (DV/DLOGDP)
C
      CALL GETVOL (TRIAL, TDIST, TOTVOL, TOTNUM )
C
C  SAVE OR DISPLAY THE FINAL RESULTS
C
      WRITE(*,124) NOWSET,HOUR
      WRITE(6,124) NOWSET,HOUR
 124  FORMAT(/'  EAA Dataset # ',I3,' at time ',F8.4 )
      WRITE(*,619) 'GETVOL',STVOL,STNUM
      WRITE(*,619) 'HISTOG',HISVOL,HISNUM
      WRITE(*,619) 'INVERT',TOTVOL,TOTNUM
      WRITE(6,619) 'GETVOL',STVOL,STNUM
      WRITE(6,619) 'HISTOG',HISVOL,HISNUM
      WRITE(6,619) 'INVERT',TOTVOL,TOTNUM
 619  FORMAT(/' Method ',A6,' gives  TV = ',F7.2,'   TN = ',1P3E10.2)
      WRITE(6,*) ' '
      IF (INFO) WRITE(6,620) HEAD
 620  FORMAT(/A/A)
      WRITE(6,*) ' '
      WRITE(6,644)
 644  FORMAT(5X,'DIAMETER   START   FINAL    VOLUME   NUMBER     RATIO'
     #/    '   I    DP      CURR*   CURR*     DIST     DIST      Fi/St')
C
      DO 300 I = NMIN,NMAX
         RAT(I) = TDIST(I)/SDIST(I)
         NDIST(I) = TDIST(I)/(3.141593*DIA(I)**3/6.)
 300  CONTINUE
C
      WRITE(6,125)  (I,DIA(I),ATRIAL(I),TRIAL(I),TDIST(I),NDIST(I),
     #                  RAT(I), I=NMIN,NMAX)
 125  FORMAT ( I4, F8.4, F9.4, F8.4, F10.3, F11.1, F8.2 )
C
C  Unformatted Files         PROFILE.ZN               PROFILE.ZV
C                    1      HOUR              1       HOUR
C  Time Profiles   2-10     HISTN(K)         2-10     HISTV(K)
C   of Distribution         11-47   NDIST(I)          11-47   TDIST(I)
C   for ZPLOT               48-50   HISNUM            48      HISVOL
```

```
C                    51-53   TOTNUM          49      TOTVOL
C
C  Midpoints    2    3    4    5    6    7    8    9   10  EAA Channel = K+1
C     I         3    7   11   15   19   23   27   31   35
C  Z curve     13   17   21   25   29   33   37   41   45
C
      IF (SAVPRO) WRITE(11) HOUR,(HISTN(K),K=1,9),(NDIST(I),I=1,37),
     #                    (HISNUM(J),J=1,3),(TOTNUM(J),J=1,3)
      IF (SAVPRO) WRITE(12) HOUR,(HISTV(K),K=1,9),(TDIST(I),I=1,37),
     #                 HISVOL,TOTVOL
      IF (SAVCUR) WRITE(20) HOUR,(TARGET(K),K=1,9),(TRIAL(I),I=1,37)
C
C        DISPLAY TRUE AND CALCULATED EAA CHANNEL CURRENTS
C
      WRITE(*,621)
      WRITE(6,621)
 621  FORMAT(/'       MIDPOINT MEASURED   CALC   RATIO    DIFF' /
     #        ' CHAN    DIA    CURRENT    CURR   Ic/Im    Ic-Im')
      DO 220 K=KMIN,KMAX
         CRAT=COUT(K)/TARGET(K)
         CDIF=COUT(K)-TARGET(K)
         WRITE(*,622) K+1, DEAA(K), TARGET(K), COUT(K), CRAT, CDIF
 220     WRITE(6,622) K+1, DEAA(K), TARGET(K), COUT(K), CRAT, CDIF
 622  FORMAT(I4, F8.4 ,F10.5, F8.3, F9.3, F9.3)
      WRITE(*,*) ' '
      WRITE(6,*) ' '
C
C  OPTION TO SAVE EACH DISTRIBUTION AS DIST.# AND HIST.#
c
      IF (SAVDIS) THEN
        IF (NOWSET.LT.10) THEN
           WRITE(DFILE,201) DNAME,NOWSET
           WRITE(SFILE,201) HNAME,NOWSET
        ELSEIF (NOWSET.LT.100) THEN
           WRITE(DFILE,202) DNAME,NOWSET
           WRITE(SFILE,202) HNAME,NOWSET
        ELSE
           WRITE(DFILE,203) DNAME,NOWSET
           WRITE(SFILE,203) HNAME,NOWSET
        ENDIF
 201    FORMAT(A,'.',I1)
 202    FORMAT(A,'.',I2)
 203    FORMAT(A,'.',I3)
C
C  DIST.n will contain the following table of results:
C  Dp  Vol(SmTw)  N(SmTw)  Vol(St)  N(St)  Vol(Sim)  N(Sim)
C
        OPEN (9,FILE=DFILE,STATUS='NEW',FORM='FORMATTED')
        DO 800 I=NMIN,NMAX
           KHI = 1 + (I-2)/4
           KLO = 1 + (I-1)/4
           IF (KLO.LT.KMIN) KLO=KMIN
      IF (KHI.GT.KMAX) KHI=KMAX
```

```
          HISV = (HISTV(KHI)+HISTV(KLO))/2.
          D = SQRT(DEAA(KHI)*DEAA(KLO))
          HISN = HISV*6./3.141593/D**3
          STN = SDIST(I)*6./3.141593/DIA(I)**3
 800      WRITE(9,810) DIA(I),TDIST(I),NDIST(I),SDIST(I),STN,HISV,HISN
 810     FORMAT(1X,F7.4,2X,3(F9.4,F11.1))
         CALL CLOSE(9)
C
C  HIST.n will contain the following table of results.
C  Dp(min/max)  Vol(Sim)  N(Sim)
C
        OPEN (8,FILE=SFILE,STATUS='NEW',FORM='FORMATTED')
        DO 850 K=KMIN,KMAX
           WRITE(8,860) DEAA(K)/EAAHLF,HISTV(K),HISTN(K)
 850       WRITE(8,860) DEAA(K)*EAAHLF,HISTV(K),HISTN(K)
 860    FORMAT(1X,F7.4,2X,F10.4,F10.1)
        CALL CLOSE(8)
      ENDIF
C
C  PROCESS NEXT EAA DATA SET
C
      IF (NOWSET.LT.MAXSET .OR. MAXSET.EQ.0) GOTO 200
C
C  EAA INVERSION PROCEDURE DONE
C
 990  CALL CLOSE (6)
      IF (SAVPRO) CALL CLOSE(11)
      IF (SAVPRO) CALL CLOSE(12)
      IF (SAVCUR) CALL CLOSE(20)
      IF (JSET.GT.0) THEN
        SETS=FLOAT(JSET)
        AVLOOP=FLOAT(JLOOP)/SETS
        AVTWO=FLOAT(JTWO)/SETS
        AVTWOL=AVTWO/AVLOOP
        AVCON=100.*FLOAT(JCON)/SETS
        AVCUR=SUMCUR/SETS
        AVCUR0=BEGCUR/SETS
        AVSIG=SUMSIG/SETS
        AVSIG0=BEGSIG/SETS
        WRITE(*,995) JSET,AVLOOP,AVTWO,AVTWOL,AVCON
        WRITE(6,995) JSET,AVLOOP,AVTWO,AVTWOL,AVCON
 995    FORMAT(/10X,'Statistics for',I4,' EAA Datasets:'/
     #   ' Average Smoothing Loops = ',F7.2 /
     #   ' Average Twomey Iterations = ',F7.2 /
     #   ' Average Twomey Iterations per Loop = ',F7.2 /
     #   ' Percentage of Datasets meeting TOLERANCE = ',F6.1,' %' )
        WRITE(*,996) AVSIG0,AVSIG,AVCUR0,AVCUR
        WRITE(6,996) AVSIG0,AVSIG,AVCUR0,AVCUR
 996    FORMAT(' Average Initial Sigma (del/tol) = ',F12.3 /
     #   ' Average Final Sigma = ',F12.3 /
     #   ' Average Initial Curvature = ',1PE11.3 /
     #   ' Average Final Curvature = ',1PE11.3 /)
      ELSE
```

```
        WRITE(*,997)
        WRITE(6,997)
 997    FORMAT(/' No Statistics Available !?'/)
      END IF
      STOP 'EAA Inversion Program THREATS Done.'
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE INPUT ( NOWSET, IERR )
C
C  THIS SUBROUTINE READS THE INPUT PARAMETERS FROM TWO FILES,
C  UNIT 2 FOR RAW EAA DATA AND UNIT 3 FOR TOLERANCES.
C
C----------------------- *.INP -------------------------------------
C1>  A70            Text to Identify Dataset
C2>  *                      ICHAN,LCHAN,KDIF(1=delta currents},MAXSET
C3+> 11F7.0                 HOUR, TARGET(K),K=KMIN,KMAX
C3a> 9F7.0          DG(3),SG(3),VM(3) {trimodal distribution}
C  Line 3 is repeated MAXSET times.  Line a used if KDIF<0
C
C----------------------- *.TOL -------------------------------------
C1> A70                     Text to Identify Tolerances
C2> *                       ICHAN,LCHAN {first & last EAA channels to use}
C3> 10F7.0          RFIT(K) {relative tolerance, -1. repeats}
C4> 10F7.0          AFIT(K) {absolute tolerance, -1. repeats}
C5> 4I7,F7.0                MAXTWO,MAXSMO,ISMAX,MINSMO,SMTOL
C6> 2I7,3F7.0               NDZERO,IGNORE,CMIN,TNOISE,DROP
C
C  IERR IS RETURNED NON-ZERO IF PARAMETERS ARE NOT IN THE PROPER RANGE
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /CHAN/ KDIV,MAG
      COMMON /FLAG/ IPRNT,INFO,INFOT
      COMMON /TEST/ DG(3),SG(3),VM(3)
      COMMON /INFO/ HOUR,MAXSET,KFIRST,KLAST,OFILE,HEAD(2)
      COMMON /SET/ TARGET(10),CTOL(10),RFIT(10),AFIT(10)
      COMMON /INVRT/ MAXTWO,MAXSMO,ISMAX,MINSMO,KDIF,SMTOL
      COMMON /DROP/ NDZERO,IGNORE,CMIN,TNOISE,DROP
      COMMON /STATS/ JSET,JLOOP,JTWO,JCON,SUMSIG,BEGSIG,SUMCUR,BEGCUR
      CHARACTER*20 OFILE,EFILE,TFILE
      CHARACTER*70 HEAD
      LOGICAL INFO,INFOT
      IERR = 0
      IF (NOWSET.LE.0) THEN
        KMIN=0
        KMAX=0
        MAXTWO=0
  MAXSMO=0
  ISMAX=0
        MINSMO=0
  KDIF=0
  SMTOL=0.
```

```
   NDZERO=0
        IGNORE=0
        CMIN=0.
        TNOISE=0.
        DROP=0.
        JSET=0
        JLOOP=0
        JTWO=0
        JCON=0
        SUMSIG=0.
        BEGSIG=0.
        SUMCUR=0.
        BEGCUR=0.
        WRITE(*,50) 'Enter EAA Tolerances Filename [EAA.TOL]: '
        READ (*,55) TFILE
        WRITE(*,50) 'Enter EAA Input Data Filename [EAA.INP]: '
        READ (*,55) EFILE
        WRITE(*,50) 'Enter Output Filename or PRN: [EAA.OUT]: '
        READ (*,55) OFILE
  50    FORMAT(1X,A\)
  55    FORMAT(A)
        IF (TFILE.EQ.' ') TFILE='EAA.TOL'
        IF (EFILE.EQ.' ') EFILE='EAA.INP'
        IF (OFILE.EQ.' ') OFILE='EAA.OUT'
        OPEN(2,FILE=EFILE,STATUS='OLD')
        OPEN(3,FILE=TFILE,STATUS='OLD')
        OPEN(6,FILE=OFILE,STATUS='NEW')
        WRITE (6,101) TFILE,EFILE,OFILE
 101    FORMAT (2X, 3(A20,2X) )
        READ (2,104) HEAD(1)
        READ (3,104) HEAD(2)
 104    FORMAT (A70)
 105    FORMAT (A70/A70)
        WRITE (6,105) HEAD
        WRITE (*,105) HEAD
        READ (2,*) ICHAN,LCHAN,KDIF,MAXSET
C
C        TEST FOR TOO MANY OR TOO FEW CHANNELS
C
        IF (LCHAN .GT. 11) THEN
          WRITE (6,* ) ' TOO MANY CHANNELS', LCHAN
        ELSEIF (ICHAN .LT. 2) THEN
          WRITE (6,* ) ' TOO FEW CHANNELS', ICHAN
        ELSE
          GOTO 100
        ENDIF
        IERR = 1
        GOTO 900
C
C        READ IN *.TOL              Note K is EAA Channel minus one.
C
 100    KFIRST=ICHAN-1
        KLAST=LCHAN-1
        READ (3,*) KMIN,KMAX
        KMIN=KMIN-1
   KMAX=KMAX-1
        IF (KMIN.LE.KFIRST .OR. KMIN.GE.KLAST) KMIN=KFIRST
        IF (KMAX.LE.KFIRST. OR. KMAX.GE.KLAST) KMAX=KLAST
        NMIN =  1 + (KMIN-1)*MAG
        NMAX = 41 - (10-KMAX)*MAG
        IF (KDIF.EQ.0) THEN
           NMAX=NMAX-MAG
           KMAX=KMAX-1
        END IF
C
C        FIT TOLERANCE FOR EACH STAGE
C
        READ (3,300) (RFIT(K),K=KMIN,KMAX)
        READ (3,300) (AFIT(K),K=KMIN,KMAX)
 300    FORMAT(10F7.0)
        READ (3,180,END=190) MAXTWO,MAXSMO,ISMAX,MINSMO,SMTOL
        READ (3,185,END=190) NDZERO,IGNORE,CMIN,TNOISE,DROP
 180    FORMAT(4I7,F7.0)
 185    FORMAT(2I7,3F7.0)
 190    CALL CLOSE(3)
C
C  USE -1. IN RFIT OR AFIT TO USE LAST VALUE FOR REMAINDER
C  IF NO LAST VALUE, DEFAULT TO RFIT(K)=.05, AFIT(K)=.002
C
        IF (RFIT(KFIRST).LT.0.) THEN
     RFIT(KFIRST)=0.05
     RFIT(KFIRST+1)=-1.
   ENDIF
   IF (AFIT(KFIRST).LT.0.) THEN
     AFIT(KFIRST)=0.002
     AFIT(KFIRST+1)=-1.
   ENDIF
        DO 196 K=KFIRST+1,KLAST
           IF (RFIT(K).LT.0.) THEN
             DO 192 I=K,KLAST
 192                RFIT(I)=RFIT(K-1)
      END IF
           IF (AFIT(K).LT.0.) THEN
             DO 194 I=K,KLAST
 194             AFIT(I)=AFIT(K-1)
           END IF
 196    CONTINUE
C
C  SET DEFAULTS
C
      IF (MAXTWO.EQ.0) MAXTWO = 30
      IF (MAXSMO.EQ.0) MAXSMO = 10
      IF (ISMAX.EQ.0) ISMAX = 3
      IF (MINSMO.EQ.0) MINSMO = 5
      IF (SMTOL.EQ.0.) SMTOL = 1.2
      IF (CMIN.LE.0.) CMIN = 1.E-5
      IF (TNOISE.LE.O.) TNOISE = 1.E-4
      IF (DROP.LE.O.) DROP = 10.
C
      END IF
C
C           SUCCESSIVE READS ACCEPT THE FOLLOWING:
C
      IF (KDIF.GE.O) THEN
         READ(2,200,END=990) HOUR,(TARGET(K), K=KFIRST,KLAST)
 200     FORMAT( F7.4, 10(F7.4) )
      ELSE
         READ (2,300) DG,SG,VM
      END IF
      NOWSET = NOWSET + 1
      IF (NOWSET.LT.IABS(MAXSET) .OR. MAXSET.EQ.0) RETURN
 900  CALL CLOSE(2)
      RETURN
 990  IERR=-1
      GOTO 900
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE SMOOTH ( Y, NMIN, NMAX, NDZERO)
C
C     SMOOTH SMOOTHS AN ARRAY Y AS FOLLOWS:
C          new Y(I) = 0.25*Y(I-1) + 0.50*Y(I) + 0.25*Y(I+1)
C     FOR THE END POINTS, ASSUME:
C         IF NDZERO=0    Y(out-of-bounds)=Y(endpoint)
C    IF NDZERO=1    Y(out-of-bounds)=0.
C  Note that NDZERO=1 still only weakly zeros the tails.
C
C>    SUBROUTINES USED -- NONE
C
      DIMENSION Y(41)
      LM1= NMAX - 1
      PAST = 0.
      IF (NDZERO.EQ.0) PAST = Y(NMIN)
      DO 10 J = NMIN,LM1
         CURR = Y(J)
         Y(J) = .25*PAST + .5*CURR + .25*Y(J+1)
         PAST = CURR
 10   CONTINUE
      IF (NDZERO.EQ.0) THEN
                             Y(NMAX) = .25*PAST + .75*Y(NMAX)
                      ELSE
                             Y(NMAX) = .25*PAST + .50*Y(NMAX)
      END IF
      RETURN
      END
C
C-----------------------------------------------------------------------
C
```

```
      SUBROUTINE RESP ( TRIAL, COUT, RM )
C
C  RESP COMPUTES THE MEASURED CURRENTS COUT GIVEN THE SIZE
C  DISTRIBUTION IN TRIAL AND THE RESPONSE MATRIX IN RM.
C  RESP MATRIX ELEMENTS LESS THAN 5.E-4 ARE SKIPPED.
C
C  FIND RESPONSE COUT(K) FROM ASSUMED I DISTRIBUTION TRIAL
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      DIMENSION TRIAL(41),COUT(10),RM(41,10)
      DO 50 K=KMIN,KMAX
         A = 0.
         DO 20 I=NMIN,NMAX
             R=RM(I,K)
             IF (R.GT.5.E-4) A=A+R*TRIAL(I)
  20     CONTINUE
         COUT(K)=A
  50  CONTINUE
C     COUT(1)=COUT(1)-0.5*TRIAL(1) ! Not Needed as RM(1,1)=0.5
      RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE GETSIG ( SZD, TRIAL )
C
C  CONVERTS VOLUME DISTRIBUTIONS INTO SUB-CHANNEL EQUIVALENT pAMPS
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /CHAN/ KDIV,MAG
      COMMON /SIZE/ DIA(41),DEAA(10)
      DIMENSION SZD(41),TRIAL(41)
      DO 100 I = NMIN,NMAX
         D = DIA(I)
         ANUM = SZD(I)*6./(3.141593*D*D*D*KDIV)
         IF ( D .LE. .0125) THEN
            TRIAL(I) = 2.351E6 * ANUM * D**6.262
         ELSE
            TRIAL(I) = 4.264E-4 * ANUM * D**1.156
         END IF
 100  CONTINUE
      RETURN
      END
C
C------------------------------------------------------------------
C
      SUBROUTINE GETVOL ( TRIAL, TDIST, TOTVOL, TOTNUM )
C
C  CALCULATE VOLUME DISTRIBUTION FROM EQUIVALENT CURRENTS
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /CHAN/ KDIV,MAG
      COMMON /SIZE/ DIA(41),DEAA(10)
      DIMENSION TDIST(41),TRIAL(41),TOTNUM(3)
C
C  TOTNUM(1) includes all, TOTNUM(2)>.01 um, TOTNUM(3)>.02 um
C
      TOTVOL = 0.
      TOTNUM(1) = 0.
      TOTNUM(2) = 0.
      TOTNUM(3) = 0.
      DO 200 I= NMIN,NMAX
         D= DIA(I)
         IF ( D .LE. .0125) THEN
            ANUM = TRIAL(I)/( 2.351E6 * D**6.262)
         ELSE
            ANUM = TRIAL(I)/( 4.264E-4 * D**1.156)
         END IF
         VOLUM = ANUM * 3.141593 * D*D*D / 6.
         TDIST(I) = VOLUM * KDIV
         TOTVOL = TOTVOL + VOLUM
         TOTNUM(1) = TOTNUM(1) + ANUM
         IF (D.GE.0.02) TOTNUM(3) = TOTNUM(3) + ANUM
         IF (D.GT.0.011) TOTNUM(2) = TOTNUM(2) + ANUM
         IF (ABS(D-.01).LT.1.E-4) TOTNUM(2) = TOTNUM(2) + 0.5*ANUM
 200  CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE HISTO ( COUT, HISTV, HISTN, TVOL, TNUM )
C
C  THIS IS THE SIMPLE EAA DATA INVERSION METHOD, WHICH ASSUMES
C  NO CROSS-SENSITIVITY.  IT IS DISCUSSED IN THE EAA MANUAL.
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /SIZE/ DIA(41),DEAA(10)
      DIMENSION COUT(10), HISTV(10), HISTN(10), CONN(10), TNUM(3)
      DATA CONN / 9.52E6, 4.17E5, 1.67E5, 8.70E4, 4.44E4,
     #               2.41E4, 1.23E4, 6.67E3, 3.51E3, 1.8E3 /
      TVOL = 0.
      TNUM(1) = 0.
      TNUM(2) = 0.
      TNUM(3) = 0.
      DO 100 K=KMIN,KMAX
         ANUM = CONN(K) * COUT(K)
         HISTN(K) = 4. * ANUM
         TNUM(1) = TNUM(1) + ANUM
         IF (K.GE.2) TNUM(2) = TNUM(2) + ANUM
         IF (K.GE.3) TNUM(3) = TNUM(3) + ANUM
         AVOL = 3.141593 * ANUM * DEAA(K)**3 / 6.
         HISTV(K) = 4. * AVOL
         TVOL = TVOL + AVOL
 100  CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE INVERT ( TRIAL, COUT, RI, RA )
C
C  THIS IS THE AUTOMATIC EAA INVERSION ROUTINE (DRIVER).
C      INVERT CALLS THE TWOMEY ROUTINE AND CONTROLS THE SMOOTHING
C      AND INTERMEDIATE OUTPUT.  RETURNS WITH SOLUTION.
C
C>    SUBROUTINES USED -- RESP,CHKOUT,FITCHK,TWOMEY,SMOOTH
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /SET/ TARGET(10),CTOL(10),RFIT(10),AFIT(10)
      COMMON /FLAG/ IPRNT,INFO,INFOT
      COMMON /INVRT/ MAXTWO,MAXSMO,ISMAX,MINSMO,KDIF,SMTOL
      COMMON /DROP/ NDZERO,IGNORE,CMIN,TNOISE,DROP
      COMMON /STATS/ JSET,JLOOP,JTWO,JCON,SUMSIG,BEGSIG,SUMCUR,BEGCUR
      LOGICAL INFO,INFOT
      DIMENSION RI(41,10),TRIAL(41),COUT(10), RA(41,10)
      DIMENSION RATIO(10), CURVE(20), TLAST(41), CLAST(10)
C
C        INTIIALIZE VARIABLES
C
      NIT = 0
      LOOPS = 0
      WRITE(6,106) MAXTWO,MAXSMO,ISMAX,NDZERO,SMTOL
 106  FORMAT(' MAXTWO=',I3,'  MAXSMO=',I3,'  ISMAX=',I3,
     #        '  NDZERO=',I2,'  SMTOL=',F7.3 )
C
C        DO LOOP TO DO TWOMEY-SMOOTHING UP TO 9 (MAXSMO) TIMES
C
      DO 60 NS = 1, MAXSMO
        ISM = 0
        IF (NS.EQ.1) GOTO 40
C
C  NOISE DISCRIMINATION
C
        IF (IGNORE.NE.0)  CALL DISCRM
     #          (TRIAL, NMIN, NMAX, TNOISE, DROP, IGNORE)
C
C        SMOOTH UNTIL DIFFERENCES ARE LARGE ENOUGH, MAX 1 (OR 5) TIMES
C
  20    CALL SMOOTH (TRIAL, NMIN, NMAX, NDZERO)
        ISM = ISM + 1
  40    CALL RESP ( TRIAL, COUT, RI )
        CALL FITCHK ( COUT, RATIO, SIGMA )
        IF ( SIGMA.LT.SMTOL .AND. ISM.LT.ISMAX .AND. NS.GT.1) GOTO 20
C
C        END OF INNER SMOOTHING LOOP
C
        LOOPS = LOOPS + 1
        ID = 2
        CALL CHKOUT(ID,LOOPS,0,NIT,ISM,TRIAL,RATIO,SIGMA,CURVAT)
        IF (LOOPS.EQ.1) THEN
            BEGSIG=BEGSIG+SIGMA
            BEGCUR=BEGCUR+CURVAT
        END IF
C
C        RUN TWOMEY TO CORRECT FOR SMOOTHING.  IT ITERATIONS DONE.
C
        CALL TWOMEY ( TRIAL, COUT, RI, RA, IT, RATIO, SIGMA )
C
        NIT = NIT + IT
        ID = 1
        CALL CHKOUT(ID,LOOPS,IT,NIT,0,TRIAL,RATIO,SIGMA,CURVAT)
        CURVE(NS) = CURVAT
C
C  OPTION TO DO MAXSMO SMOOTHING LOOPS, OVERRIDE AUTOMATIC EXIT
C
        IF (LOOPS.LT.MINSMO) GOTO 50
C
C        STOP SMOOTHING LOOP IF CURVATURE INCREASES
C
        IF (NS.GT.2 .AND. CURVAT .GT. CURVE(NS-1)) GOTO 70
C
C         QUIT IF LESS THAN .025 DECREASE IN LAST 2 ITERATIONS
C
        IF (NS.GT.3 .AND. CURVAT .GE. .975*CURVE(NS-2)) GOTO 90
C
C         SAVE THIS TRIAL AND COUT IN TLAST AND CLAST
C
  50    DO 59 I = NMIN,NMAX
  59       TLAST(I) = TRIAL(I)
        DO 56 K = KMIN,KMAX
  56       CLAST(K) = COUT(K)
  60  CONTINUE
C
      WRITE(6,65) MAXSMO
      WRITE(*,65) MAXSMO
  65  FORMAT(' *** MAXIMUM ',I2,' TWOMEY-SMOOTHING LOOPS DONE.')
      GOTO 100
C
C         USE PRIOR TRIAL AND CHAN CURRENTS INSTEAD OF NEW ONES
C
  70  DO 72 MM = NMIN,NMAX
  72     TRIAL(MM) = TLAST(MM)
      DO 74 MM = KMIN,KMAX
  74     COUT(MM) = CLAST(MM)
      WRITE(*,*) ' Curvature Increased so PRIOR TRIAL USED'
      WRITE(6,*) ' PRIOR TRIAL USED'
C
  90  WRITE(6,95) LOOPS
  95  FORMAT(/' *** INVERSION USED ',I2,' TWOMEY-SMOOTHING LOOPS.'/)
 100  WRITE(6,105) (CURVE(J), J=1,LOOPS)
 105  FORMAT(' CURVAT ',8F9.5)
```

```
      JSET=JSET+1
      IF (SIGMA.LE.1.0) JCON=JCON+1
      JLOOP=JLOOP+LOOPS
      JTWO=JTWO+NIT
      SUMSIG=SUMSIG+SIGMA
      SUMCUR=SUMCUR+CURVAT
      RETURN
      END
C
C------------------------------------------------------------------
C
      SUBROUTINE TWOMEY ( TRIAL, COUT, RI, RA, IT, RATIO, SIGMA )
C
C        TWOMEY DOES THE TWOMEY ITERATION UNTIL SIGMA IS <1;IF THE RES
C     MATRIX ELEMENT IS LESS THAN .005 THAT CORECTION IS SKIPPED. THE
C     ITERATION FOR A STAGE IS SKIPPED IF THE TRIAL & TRUE CURRENTS ARE
C     WITHIN THE CHANNEL FIT TOLERANCE.
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /SET/ TARGET(10),CTOL(10),RFIT(10),AFIT(10)
      COMMON /INVRT/ MAXTWO,MAXSMO,ISMAX,MINSMO,KDIF,SMTOL
      DIMENSION TRIAL(41), RATIO(10),RI(41,10), RA(41,10)
      DIMENSION COUT(10)
C
C        DO UP TO 30 (MAXTWO) TWOMEY ITERATIONS
C    ITERATE ON CHANNEL INDEX K FIRST
C      SKIP CHANNEL IF IT IS WITHIN TOLERANCE
C          ADJUST TRIAL FOR EACH SIZE WITH RA > .005
C    CALCULATE NEW EXPECTED EAA CURRENT FOR UPDATED TRIAL
C    CHECK AGREEMENT WITH ACTUAL EAA CURRENTS
C    DISPLAY PROGRESS ON SCREEN
C  CONTINUE TWOMEY ITERATIONS IF AGREEMENT INADEQUATE, SIGMA>1
C
      DO 50 J = 1, MAXTWO
         IT = J
         DO 40 K = KMIN,KMAX
            CERR = TARGET(K) - COUT(K)
C
C  COULD TRY TO MAKE SUFFICIENTLY GOOD FIT BETTER BY
C  UNCONDITIONALLY DOING THE NEXT IF BLOCK, BUT THIS WOULD
C  DECREASE THE SMOOTHNESS.
C
            IF ( ABS(CERR) .GE. CTOL(K) ) THEN
               A  = RATIO(K)
               DO 30 I = NMIN, NMAX
                  B = RA(I,K)
                  IF ( B .GT. .005) THEN
                    TRIAL(I) = TRIAL(I)*(1.+A*B)
C                   WRITE(6,100) I,K,TRIAL(I), A, B
C100                   FORMAT( 2I5, 3F9.4)
                  END IF
 30            CONTINUE
            END IF
 40      CONTINUE
C
         CALL RESP ( TRIAL, COUT, RI)
         CALL FITCHK ( COUT, RATIO, SIGMA )
C        WRITE (6, 105) (RATIO(K), K=KMIN,KMAX)
C105     FORMAT (' RATIOS', 10F7.3)
         IF (SIGMA .LT. 1.) THEN
C           WRITE(*,140) J
C140        FORMAT(' Tw Pass=',I2)
            RETURN
         END IF
C        IF ( MOD(J,5) .EQ. 0 ) WRITE (*,145) J
C145     FORMAT( ' Tw Pass ',I2,' '\)
  50  CONTINUE
C
      WRITE(6,*) ' *** FIT NOT MET AFTER MAXIMUM TWOMEY ITERATIONS'
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE FITCHK ( COUT, RATIO, SIGMA )
C
C        FITCHK COMPUTES A FRACTIONAL DISCREPANCY IN THE
C  DESIRED SIGNAL TARGET FROM THE CALCULATED SIGNAL COUT,
C  AND FINDS A NORMALIZED ERROR PARAMETER SIGMA.
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /SET/ TARGET(10),CTOL(10),RFIT(10),AFIT(10)
      DIMENSION COUT(10),RATIO(10)
      NCHAN = KMAX-KMIN+1
      SIGMA = 0.
      DO 10 K = KMIN,KMAX
         RATIO(K) = TARGET(K)/COUT(K) - 1.
         A = (COUT(K)-TARGET(K)) / CTOL(K)
         SIGMA = SIGMA + A*A
  10  CONTINUE
      SIGMA = SQRT( SIGMA/NCHAN )
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE CHKOUT(ID,LOOPS,IT,NIT,ISM,TRIAL,RATIO,SIGMA,CURVAT)
C
C     CHKOUT PRINTS INTERMEDIATE INVERSION RESULTS AND GETS
C     CURVATURE PARAMETER. IPRNT=1 CAUSES TRIAL TO BE PRINTED AFTER
C     TWOM, 2=AFTER SMOOTH ONLY
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /FLAG/ IPRNT,INFO,INFOT
      LOGICAL INFO,INFOT
      DIMENSION RATIO(10),TRIAL(41)
```

```
C
C         CALCULATE CURVATURE PARAMETER (MUST SKIP ENDPOINTS)
C
      CURVAT = 0.
      DO 10 I = NMIN+1, NMAX-1
         A = TRIAL(I)
         CURVAT = CURVAT + ABS( A+A - TRIAL(I-1) - TRIAL(I+1) )
 10   CONTINUE
      NCOUNT = NMAX-NMIN-1
      CURVAT = CURVAT / NCOUNT
C
C  PRINT OUT STATUS OF TWOMEY-SMOOTHING SCHEME
C
      IF (INFOT) WRITE(6,90) LOOPS,IT,NIT,ISM,SIGMA,CURVAT
      WRITE(*,90) LOOPS,IT,NIT,ISM,SIGMA,CURVAT
  90  FORMAT(' Loop',I3,'   Tw=',I3,' =>',I4,'   Sm=',I3,
     #              '   SIGMA=',F8.2,'   CURV=',F9.5)
C     IF (INFOT.AND.IPRNT .GE. 0 ) WRITE(6,95) (RATIO(K), K = KMIN,KMAX)
  95  FORMAT(10F8.3)
C
C12   TYPE 'TYPE INTEGER,1 PRINTS AFTER TWOM,2 AFTER SMOOTH,>9 ALL'
C     READ(5,*) IPRNT
C
      IF (INFOT.AND.(ID.EQ.IPRNT.OR.IPRNT.GE.10)) THEN
C         PRINT TRIAL IN 4 COLUMNS
        NCOL = 4
        NLIN = (NMAX-NMIN)/NCOL + 1
        DO 200 I = 1,NLIN
           JJ=NCOL-1
           IF (I+JJ*NLIN.GT.NMAX) JJ=JJ-1
           WRITE(6,110) (I+J*NLIN,TRIAL(I+J*NLIN),J=0,JJ)
 110       FORMAT(5(I3,F9.4,4X))
 200    CONTINUE
      END IF
 900  RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE DISCRM (TRIAL, NMIN, NMAX, TNOISE, DROP, IGNORE)
C
C  NOISE DISCRIMINATION ROUTINE, BEFORE SMOOTHING
C  IF PART OF TRIAL GOES BELOW TNOISE, IT IS PUSHED TOWARDS
C   ZERO AT A RATE OF DROP PER INTERVAL (OR DROP**4.816 PER
C   FACTOR OF TWO IN DIAMETER).
C  IF TNOISE IS LESS THAN CTOL/4. THEN THE TWOMEY ROUTINE
C   ISN'T LIKELY TO FIGHT TO RAISE THE FALLEN CURVE;
C   OTHERWISE TMOMEY MAY NOT ACCEPT WHAT DISCRM TRIES
C   TO DO.
C  The rationale of DISCRM goes something like this:
C    If the value of TRIAL(I) appears to be S +/- N with
C     S < N (less signal than noise), then we really
C     can't detect the signal, so we should call it zero
```

```
C     (within the constraint that our curves stay smooth).
C     This especially true when the SMOOTHING and the minimum
C     initial guess and the EAA cross-sensitivity all tend
C     to raise S above zero (so S is decidedly a high-biased
C     guess of the true signal, when we are at the tails of
C     the distribution).
C
      COMMON /FLAG/ IPRNT,INFO,INFOT
      LOGICAL INFO,INFOT
      DIMENSION TRIAL(41)
      IF (IGNORE.EQ.0) RETURN
      I = NMIN-1
 100  I = I + 1
      IF (I.GT.NMAX) GOTO 900
      IF (TRIAL(I).GE.TNOISE) GOTO 100
C
C  START OF SUB-THRESHOLD READINGS
C
      ISTAR = I
 200  I = I + 1
      IF (I.GT.NMAX) GOTO 800
      IF (TRIAL(I).LT.TNOISE) GOTO 200
C
C  END OF SUB-THRESHOLD READINS
C
      ISTOP = I - 1
      IF (ISTAR.EQ.NMIN) GOTO 700
C
C  SUB-THRESHOLD IN MIDDLE . . . IGNORE IF 1 CHANNEL WIDE OR LESS
C
      IF (IGNORE.LT.0) GOTO 100
      ISIZE = (ISTOP-ISTAR+1)
      IF (ISIZE.LE.4) GOTO 100
      NSIZE = (ISIZE / 2)
      DO 300 J = 0, NSIZE-1
        TRIAL(ISTAR+J) = TRIAL(ISTAR-1) / DROP ** (J+1)
 300    TRIAL(ISTOP-J) = TRIAL(ISTOP+1) / DROP ** (J+1)
      IF (MOD(ISIZE,2).EQ.1) THEN
        IMID=(ISTAR+ISTOP)/2
        TRIAL(IMID) = AMAX1(TRIAL(IMID+1),TRIAL(IMID-1)) / DROP
      END IF
      WRITE(*,400) ISTAR,ISTOP,DROP
      IF (INFO) WRITE(6,400) ISTAR,ISTOP,DROP
 400  FORMAT('  DISCRM reduced TRIAL(',I2,'-',I2,') by ',F5.1)
      GOTO 100
C
C  TAIL AT START
C
 700  ISIZE=ISTOP-NMIN+1
      DO 750 J=0,ISIZE-1
 750    TRIAL(ISTOP-J) = TRIAL(ISTOP+1) / DROP ** (J+1)
      WRITE(*,400) NMIN,ISTOP,DROP
      IF (INFO) WRITE(6,400) NMIN,ISTOP,DROP
      GOTO 100
C
C  TAIL AT END
C
 800  ISIZE=NMAX-ISTAR+1
      IF (ISTAR.EQ.NMIN) THEN
        DO 820 J=NMIN,NMAX
 820        TRIAL(J)=1.E-10
      ELSE
        DO 850 J=0,ISIZE-1
 850        TRIAL(ISTAR+J) = TRIAL(ISTAR-1) / DROP ** (J+1)
      END IF
      WRITE(*,400) ISTAR,NMAX,DROP
      IF (INFO) WRITE(6,400) ISTAR,NMAX,DROP
 900  RETURN
      END
C
C-------------------------------------------------------------------
C
      SUBROUTINE INIT ( A, N, CONST )
C
C         INIT MERELY INITIALIZES ARRAY A TO CONST.
C
      DIMENSION A(N)
      DO 10 J = 1,N
  10    A(J) = CONST
      RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE LOGNRM ( NDIS, SIZD )
C
C        LGNORM CALCULATES A LOGNORMAL SIZE DISTRIBUTION, SIZD, USING
C     DIAMETER DG AND DEVIATION SG AND VOLUME VM INDICATED BY NDIS
C
C>    SUBROUTINES USED -- NONE
C
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      COMMON /TEST/ DG(3),SG(3),VM(3)
      COMMON /SIZE/ DIA(41),DEAA(10)
      DIMENSION SIZD(41)
      SDLOG = ALOG ( SG(NDIS) )
C  Normal distribution prefactor is 1./SQRT(2.*PI)
C  Note ln(10) pops up because we use dV/dlog10(dp)
      ANORM = 0.9186 * VM(NDIS) / SDLOG
      DPMEAN = DG(NDIS)
      WRITE(*,25) NDIS,DPMEAN,SG(NDIS),VM(NDIS)
  25  FORMAT(' LOGNORMAL DISTRIBUTION #',I2,':  Dp=',F9.4,
     #          '  Sg=',F9.4,'  Vm=',F9.4)
      DO 50 I = NMIN,NMAX
         Z = ALOG ( DIA(I)/DPMEAN ) / SDLOG
         A = ANORM * EXP ( - Z * Z / 2. )
         SIZD(I) = SIZD(I) + A
 50   CONTINUE
      RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE NONEG(CURR)
      COMMON /SUBS/ KMIN,KMAX,NMIN,NMAX
      REAL CURR(10)
C
C  Negativity compensation algorithm.
C  Insists that all differences be positive, or at least zero.
C  If CURR(K)<0. then will attempt to split the negativity with
C   the neighboring points, spreading out as far as necessary
C   to eliminate the impossible negative signal.  An endpoint
C   is a perfect sink.  At each distance from the negative
C   signal, the algorithm attempts to split the negative
C   burden evenly, and if still not satisfied will take whatever
C   is needed from the remaining positive signal (never driving
C   any signal negative), and spread out further, bilaterally,
C   from the negative source if necessary.
C
C  J = distance from source K (1,2,3 . . .)
C  COVER = amount of negative current still needing to be covered
C  UP = maximum signal that up channel K+J could cover
C  DOWN = maximum signal that down channel K-J could cover
C
      DO 500 K=KMIN,KMAX
         IF (CURR(K).LT.0.) THEN
       COVER=-CURR(K)
           J=1
200        KD=K-J
           KU=K+J
           DOWN=COVER
       UP=COVER
           IF (KD.GE.KMIN) DOWN=CURR(KD)
           IF (KU.LE.KMAX) UP=CURR(KU)
           IF (DOWN.LT.0.) DOWN=0.
           IF (UP.LT.0.) UP=0.
           PLAY=DOWN+UP
           IF (PLAY.GE.COVER) GOTO 400
           COVER=COVER-PLAY
           IF (KU.LE.KMAX .AND. CURR(KU).GT.0.) CURR(KU)=0.
           IF (KD.GE.KMIN .AND. CURR(KD).GT.0.)  CURR(KD)=0.
           J=J+1
           IF (J.LE.4) GOTO 200
           WRITE(*,*) 'NONEG WARNING:  Cannot Cover Difference!'
           GOTO 450
400        IF (UP.GE.0.5*COVER .AND. DOWN.GE.0.5*COVER) THEN
              IF (KD.GE.KMIN) CURR(KD)=CURR(KD)-0.5*COVER
              IF (KU.LE.KMAX) CURR(KU)=CURR(KU)-0.5*COVER
           ELSE IF (UP.LT.0.5*COVER) THEN
```

```
               IF (KU.LE.KMAX .AND. CURR(KU).GT.O.) CURR(KU)=0.
               IF (KD.GE.KMIN) CURR(KD)=CURR(KD)-COVER+UP
            ELSE IF (DOWN.LT.0.5*COVER) THEN
               IF (KD.GE.KMIN .AND. CURR(KD).GT.O.) CURR(KD)=0.
               IF (KU.LE.KMAX) CURR(KU)=CURR(KU)-COVER+DOWN
            ELSE
               STOP 'LOGIC ERROR IN NONEG!'
            ENDIF
 450        CURR(K)=0.
         END IF
C     WRITE(*,222) (CURR(J),J=KMIN,KMAX)
C222  FORMAT(' Curr ',9F7.4)
 500  CONTINUE
      RETURN
      END
      PROGRAM SELECT
C
C  MANIPUATES INVERTED EAA DISTIBUTION AT USER-SELECTED TIME
C
C  THE INPUT FILE IS THE CURRENT (.ZC) UNFORMATTED DATA FILE
C
C  SELECT will find the user-selected time, optionally smoothing
C  its EAA data, and output data to HIST.# & DIST.# files,
C  optionally creating number/volume/size summary (.ES) and
C  current profile (.ZV and .ZN) files.
C
C         Programmed JAN-86 by DRW for the AT     V2.1
C
C  See THREATS.DOC and EPLOT.DOC for more information.
C
C       DATA FILES:
C
C  20     CFILE   CURRENT.ZC      Inverted Currents (INPUT)
C  11     NFILE   PROFILE.ZN      Inverted Number
C  12     VFILE   PROFILE.ZV      Inverted Volume
C        8 HFILE   HIST.#          Histogram Distribution
C        9 DFILE   DIST.#          Inverted Distribution
C  25     SFILE   SEE.ES          Totals and average Dps.
C
C
      COMMON /SIZE/ DIA(37),DEAA(9)
      COMMON /STORE/ TIME(200),CURR(9,200),CU(37,200)
      COMMON /WHERE/ NOW,NEXT
      REAL CSET(9),TRIAL(37)
      REAL PRAW(9),POUT(37)
      REAL VDIST(37),NDIST(37),TOTNUM(3),XD(37)
      REAL HISTV(9),HISTN(9),HISNUM(3)
      CHARACTER*20 DFILE,HFILE,NFILE,VFILE,CFILE,SFILE
      CHARACTER*4 DNAME,HNAME
      CHARACTER*1 ASK
      LOGICAL SAVPRO,SAVDIS,SAVSUM
C
C  SELECT uses Channels 2-10 of the EAA (fixed by default)
C
      DATA NMIN,NMAX / 1,37 /
      DATA KMIN,KMAX / 1,9 /
      DATA KDIV / 16 /
      EAAHLF = 10.**0.125
C
C        SET DIAMETERS      Beginning with Channel 2 of EAA
C
      DIA(1) = 0.005623413
      DO 100 I = 2,37
 100     DIA(I) = DIA(I-1) * 1.154782
      DEAA(1) = 0.007498942
      DO 110 K = 2,9
 110     DEAA(K) = DEAA(K-1) * 1.778279
```

```
      DO 120 I = 1,37
 120     XD(I) = FLOAT(I-37)/16.
C
C          WHAT INFO SHOULD BE SAVED FROM THE INVERSION?
C
      WRITE(*,190)
 190  FORMAT(/20X,'*** SELECT - EAA PROFILE HANDLER ***'/)
      WRITE(*,200) 'NAME OF CURRENT FILE [CURRENT.ZC] : '
 200  FORMAT(/1X,A\)
      READ(*,202) CFILE
 202  FORMAT(A)
      IF (CFILE.EQ.' ') CFILE='CURRENT.ZC'
C
      DNAME='DIST'
      HNAME='HIST'
      NFILE='PROFILE.ZN'
      VFILE='PROFILE.ZV'
      SFILE='SEE.ES'
      OPEN (20,FILE=CFILE,STATUS='OLD',FORM='UNFORMATTED')
C
      WRITE(*,200) 'Shall N & V profiles be created? [N] '
      READ(*,666) ASK
 666  FORMAT(A1)
      SAVPRO=.FALSE.
      IF (ASK.EQ.'Y' .OR. ASK.EQ.'y') SAVPRO=.TRUE.
      IF (SAVPRO) THEN
        OPEN (11,FILE=NFILE,STATUS='NEW',FORM='UNFORMATTED')
        OPEN (12,FILE=VFILE,STATUS='NEW',FORM='UNFORMATTED')
      ENDIF
      WRITE(*,200) 'Shall EAA Summary (.ES) be created? [Y] '
      READ(*,666) ASK
      SAVSUM=.TRUE.
      IF (ASK.EQ.'N' .OR. ASK.EQ.'n') SAVSUM=.FALSE.
      IF (SAVSUM) THEN
        OPEN (25,FILE=SFILE,STATUS='NEW',FORM='FORMATTED')
      ENDIF
C
C  Unformatted Files of the EAA Size Distribution with Time
C
C  CURRENT.ZC                 PROFILE.ZN              PROFILE.ZV
C  1        HOUR             1        HOUR           1        HOUR
C  2-10     CSET(K)          2-10     HISTN(K)       2-10     HISTV(K)
C  11-47    TRIAL(I)         11-47    NDIST(I)       11-47    VDIST(I)
C                            48-50    HISNUM         48       HISVOL
C                            51-53    TOTNUM         49       TOTVOL
C
C  Midpoints    2   3   4   5   6   7   8   9   10  EAA Channel = K+1
C     I         3   7  11  15  19  23  27  31   35
C  Z curve     13  17  21  25  29  33  37  41   45
C
C  ALL TIMES WILL BE DIFFERENCES FROM START TIME
C
      WRITE(*,200) 'Run START Time (Hours; 0. for real time): '
      READ(*,*) START
      NOW = 0
C
C          READ IN EAA DATASET
C
 300  READ(20,END=400,ERR=390) HOUR,(CSET(K),K=1,9),(TRIAL(I),I=1,37)
      HOUR=HOUR-START
      NOW = NOW + 1
C     WRITE(*,305) NOW,HOUR
C305  FORMAT(' TIME # ',I4,' = ',F7.3)
      IF (SAVPRO.OR.SAVSUM) THEN
         CALL HISTO ( CSET, HISTV, HISTN, HISVOL, HISNUM )
         CALL GETDIS ( TRIAL, VDIST, NDIST, TOTVOL, TOTNUM )
      ENDIF
      IF (SAVPRO) THEN
         WRITE(11) HOUR,(HISTN(K),K=1,9),(NDIST(I),I=1,37),
     #                     (HISNUM(J),J=1,3),(TOTNUM(J),J=1,3)
         WRITE(12) HOUR,(HISTV(K),K=1,9),(VDIST(I),I=1,37),
     #                   HISVOL,TOTVOL
      ENDIF
      IF (SAVSUM) THEN
         CALL FINDAV(VDIST,NDIST,DPVAV,DPNAV)
         CALL PEAK(VDIST,DVMODE)
         CALL PEAK(NDIST,DNMODE)
         CALL LNORML(VDIST,XD,DVLOG,SG,AGREE)
         WRITE(25,290) HOUR,TOTNUM(2),TOTVOL,DNMODE,DPNAV,DPVAV,
     #                     DVMODE,DVLOG,SG,AGREE
 290     FORMAT(1X,F7.3,F9.0,F8.3,5F8.4,F6.3,F6.3)
C
C           Formatted EAA Summary (.ES) File format:
C
C       1     2    3    4        5       6        7       8        9      10
C     TIME   NT   VT   DPnmod DPnav   DPvav      DPvmod  DPvlav   SG    AGREE
C
      ENDIF
      TIME(NOW)=HOUR
      DO 310 K=1,9
 310     CURR(K,NOW)=CSET(K)
      DO 320 I=1,37
 320     CU(I,NOW)=TRIAL(I)
      IF (NOW.LT.200) GOTO 300
C
 390  WRITE(*,*) ' TROUBLE -- too much data or file error'
C
 400  CONTINUE
      ENDTIM=HOUR
      WRITE(*,*) NOW,' EAA Datasets Read up to Time ',ENDTIM
      CALL CLOSE (20)
      CALL CLOSE (11)
      CALL CLOSE (12)
C
C  ALL EAA CURRENT DATA HAS BEEN READ IN
C
```

```
C  SAVE OR DISPLAY THE FINAL RESULTS
C
      WRITE(*,200) 'AVERAGING Interval (Hours; 0. nearest pt.): '
      READ(*,*) TINT
      WRITE(*,200) 'Time Into Run for FIRST Plot (Hours): '
      READ(*,*) BEGIN
      WRITE(*,200) 'Time STEP Between Plots (Hours): '
      READ(*,*) TSTEP
C
C  Initialize Plotting Loop
C
      NEXT=1
      PTIME=BEGIN
 500  CALL AVERAG(PTIME,TINT,PRAW,POUT)
C
C  Note if there is no data near PTIME, AVERAG will use the
C   first time after PTIME to provide data, rather than exit.
C
      CALL HISTO ( PRAW, HISTV, HISTN, HISVOL, HISNUM )
      CALL GETDIS ( POUT, VDIST, NDIST, TOTVOL, TOTNUM )
C
C  OPTION TO SAVE EACH DISTRIBUTION AS DIST.# AND HIST.#
C
      IF (NEXT.LT.10) THEN
         WRITE(DFILE,401) DNAME,NEXT
         WRITE(HFILE,401) HNAME,NEXT
      ELSEIF (NEXT.LT.100) THEN
         WRITE(DFILE,402) DNAME,NEXT
         WRITE(HFILE,402) HNAME,NEXT
      ELSE
         WRITE(DFILE,403) DNAME,NEXT
         WRITE(HFILE,403) HNAME,NEXT
      ENDIF
 401  FORMAT(A,'.',I1)
 402  FORMAT(A,'.',I2)
 403  FORMAT(A,'.',I3)
C
C  DIST.n will contain the following table of results:
C  Dp  Vol(SmTw)  N(SmTw)  Vol(St)  N(St)  Vol(Sim)  N(Sim)
C
      OPEN (9,FILE=DFILE,STATUS='NEW',FORM='FORMATTED')
      DO 800 I=NMIN,NMAX
         KHI = 1 + (I-2)/4
         KLO = 1 + (I-1)/4
         IF (KLO.LT.KMIN) KLO=KMIN
         IF (KHI.GT.KMAX) KHI=KMAX
         HISV = (HISTV(KHI)+HISTV(KLO))/2.
         HISN = (HISTN(KHI)+HISTN(KLO))/2.
 800     WRITE(9,810) DIA(I),VDIST(I),NDIST(I),1.,1.,HISV,HISN
 810  FORMAT(1X,F7.4,2X,3(F9.4,F11.1))
      CALL CLOSE(9)
C
C  HIST.n will contain the following table of results.
```

```
C  Dp{min/max}  Vol(Sim)  N(Sim)
C
      OPEN (8,FILE=HFILE,STATUS='NEW',FORM='FORMATTED')
      DO 850 K=KMIN,KMAX
         WRITE(8,860) DEAA(K)/EAAHLF,HISTV(K),HISTN(K)
 850     WRITE(8,860) DEAA(K)*EAAHLF,HISTV(K),HISTN(K)
 860  FORMAT(1X,F7.4,2X,F10.4,F10.1)
      CALL CLOSE(8)
C
C  PROCESS NEXT EAA DATA SET
C
      NEXT=NEXT+1
      PTIME=PTIME+TSTEP
      IF (PTIME.LT.ENDTIM) GOTO 500
C
C  EAA INVERSION PROCEDURE DONE
C
 990  CONTINUE
      STOP 'EAA Data Management Program SELECT Done.'
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE GETDIS ( TRIAL, VDIST, NDIST, TOTVOL, TOTNUM )
C
C  CALCULATE VOLUME DISTRIBUTION FROM EQUIVALENT CURRENTS
C
      COMMON /SIZE/ DIA(37),DEAA(9)
      REAL TRIAL(37), VDIST(37), NDIST(37), TOTNUM(3)
C
C  TOTNUM(1) includes all, TOTNUM(2)>.01 um, TOTNUM(3)>.02 um
C
      DATA NMIN,NMAX / 1 , 37 /
      DATA KDIV / 16 /
      TOTVOL = 0.
      TOTNUM(1) = 0.
      TOTNUM(2) = 0.
      TOTNUM(3) = 0.
      DO 200 I= NMIN,NMAX
         D= DIA(I)
         IF ( D .LE. .0125) THEN
            ANUM = TRIAL(I)/( 2.351E6 * D**6.262)
         ELSE
            ANUM = TRIAL(I)/( 4.264E-4 * D**1.156)
         END IF
         VOLUM = ANUM * 3.141593 * D*D*D / 6.
         VDIST(I) = VOLUM * KDIV
         NDIST(I) = ANUM * KDIV
         TOTVOL = TOTVOL + VOLUM
         TOTNUM(1) = TOTNUM(1) + ANUM
         IF (D.GE.0.02) TOTNUM(3) = TOTNUM(3) + ANUM
         IF (D.GT.0.011) TOTNUM(2) = TOTNUM(2) + ANUM
         IF (ABS(D-.01).LT.1.E-4) TOTNUM(2) = TOTNUM(2) + 0.5*ANUM
 200  CONTINUE
      RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE HISTO ( COUT, HISTV, HISTN, TVOL, TNUM )
C
C  THIS IS THE SIMPLE EAA DATA INVERSION METHOD, WHICH ASSUMES
C  NO CROSS-SENSITIVITY.  IT IS DISCUSSED IN THE EAA MANUAL.
C
      COMMON /SIZE/ DIA(37),DEAA(9)
      DIMENSION COUT(10), HISTV(10), HISTN(10), CONN(10), TNUM(3)
      DATA KMIN,KMAX / 1, 9 /
      DATA CONN / 9.52E6, 4.17E5, 1.67E5, 8.70E4, 4.44E4,
     #               2.41E4, 1.23E4, 6.67E3, 3.51E3, 1.8E3 /
      TVOL = 0.
      TNUM(1) = 0.
      TNUM(2) = 0.
      TNUM(3) = 0.
      DO 100 K=KMIN,KMAX
         ANUM = CONN(K) * COUT(K)
         HISTN(K) = 4. * ANUM
         TNUM(1) = TNUM(1) + ANUM
         IF (K.GE.2) TNUM(2) = TNUM(2) + ANUM
         IF (K.GE.3) TNUM(3) = TNUM(3) + ANUM
         AVOL = 3.141593 * ANUM * DEAA(K)**3 / 6.
         HISTV(K) = 4. * AVOL
         TVOL = TVOL + AVOL
 100  CONTINUE
      RETURN
      END
C
C------------------------------------------------------------------
C
      SUBROUTINE AVERAG ( PTIME, TINT, PRAW, POUT )
C
C  Finds AVERAGE EAA Currents at PTIME within Interval TINT
C     (from PTIME-TINT/2 to PTIME+TINT/2 in hours)
C
      REAL PRAW(9), POUT(37)
      COMMON /STORE/ TIME(200),CURR(9,200),CU(37,200)
      COMMON /WHERE/ NOW,NEXT
C
      PSTAR=PTIME-TINT/2.
      PSTOP=PTIME+TINT/2.
      J=1
  25  IF (TIME(J).GT.PSTAR) GOTO 100
      J=J+1
      IF (J.GT.NOW) STOP 'PTIME too large'
      GOTO 25
 100  DO 110 K=1,9
 110     PRAW(K)=CURR(K,J)
```

```
      DO 120 I=1,37
 120     POUT(I)=CU(I,J)
      KOUNT=1
 200  J=J+1
      IF (J.GT.NOW .OR. TIME(J).GT.PSTOP) GOTO 500
      DO 210 K=1,9
 210     PRAW(K)=PRAW(K)+CURR(K,J)
      DO 220 I=1,37
 220     POUT(I)=POUT(I)+CU(I,J)
      KOUNT=KOUNT+1
      GOTO 200
 500  COUNT=FLOAT(KOUNT)
      DO 510 K=1,9
 510     PRAW(K)=PRAW(K)/COUNT
      DO 520 I=1,37
 520     POUT(I)=POUT(I)/COUNT
      WRITE(*,900) NEXT,PTIME,KOUNT
 900  FORMAT(' DISTribution #',I3,' at TIME ',F7.3,' from',I3,
     #             ' DATASETS.')
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE FINDAV ( VDIST, NDIST, DPVAV, DPNAV )
      COMMON /SIZE/ DIA(37),DEAA(9)
      REAL VDIST(37),NDIST(37)
      DATA NMIN,NMAX / 5,37 /
      DATA KDIV / 16 /
C
C  FIND MOMENTS OF THE SIZE DISTRIBUTION FROM THE MESH
C  This routine should be double-checked.  Both the formula
C        and the endpoints used to calculate the characteristic
C   diameter can change the result significantly, so these
C   "average" or "typical" diameters need to be explained.
C
      SUMN=0.
      SUMV=0.
      SUM1=0.
      DO 100 I=NMIN,NMAX
         SUMN=SUMN+NDIST(I)
         SUMV=SUMV+VDIST(I)
         SUM0=SUM0+NDIST(I)
         SUM1=SUM1+NDIST(I)*DIA(I)
 100  CONTINUE
      DPVAV= (6*SUMV/SUMN/3.141593)**(1./3.)
      DPNAV= SUM1/SUMN
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PEAK ( DIST, DMODE )
```

```
C
C  Finds Peak of DIST using Cubic Splines and Binary Search
C
      COMMON /SIZE/ DIA(37),DEAA(9)
      REAL DIST(37)
      REAL X(7),Y(7),C(6,3),XTRY(1),YTRY(1)
      DATA NMIN,NMAX / 7,37 /
      DATA NX,IC,M / 7,6,1 /
C
C  FIND LARGEST MESH POINT OF DISTRIBUTION
C
      TOP=0.
      MAX=1
      DO 100 I=NMIN,NMAX
         IF (DIST(I).GT.TOP) THEN
            TOP=DIST(I)
            MAX=I
         ENDIF
 100  CONTINUE
C
C  USE CUBIC SPLINE TO ESTIMATE EXACT MODE OF DISTRIBUTION
C
      ISTART=MAX-4
      IF (ISTART.LT.NMIN-1) ISTART=NMIN-1
      IF (ISTART.GT.NMAX-7) ISTART=NMAX-7
      DO 200 I=1,7
         N = ISTART + I
         X(I)=FLOAT(N)
 200     Y(I)=DIST(N)
      CALL ICSCCU ( X, Y, NX, C, IC, IER )
C
C  NOW FIND MAXIMUM WITHIN CUBIC SPLINE REGION
C  This means find where the first derivative equals 0.,
C   and where second devirative is positive.  If multiple
C   roots, we may have a problem!  To make life easier,
C   we'll only search for roots on each interval by the
C   knot with the maximum value.
C
C
      IMIN = MAX-1
      IMAX = MAX+1
      IF (IMIN.LT.NMIN) IMIN=NMIN
      IF (IMAX.GT.NMAX) IMAX=NMAX
      XMIN=FLOAT(IMIN)
      XMAX=FLOAT(IMAX)
      CALL DCSEVU(X,Y,NX,C,IC,XMIN,DMIN,1,D2MIN,1,IER)
      CALL DCSEVU(X,Y,NX,C,IC,XMAX,DMAX,1,D2MAX,1,IER)
      IF (DMIN*DMAX.GE.0. .AND. IMAX-IMIN.GT.1) THEN
         WRITE(*,*) ' TROUBLE -- Multiple Extrema Near Peak'
      ELSE
        DO 400 J=1,20
           XCEN=0.5*(XMIN+XMAX)
           CALL DCSEVU(X,Y,NX,C,IC,XCEN,DCEN,1,D2CEN,1,IER)
```

```
          IF (DCEN*DMIN.GE.0.) THEN
             XMIN=XCEN
             DMIN=DCEN
          ELSE
             XMAX=XCEN
             DMAX=DCEN
          ENDIF
 400    CONTINUE
      ENDIF
      XCEN=0.5*(XMIN+XMAX)
      MAX=INT(XCEN)
      DCEN=XCEN-MAX
      DMODE=DIA(MAX)*10.**(DCEN/16.)
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE LNORML(VDIST,XD,DVLOG,SG,AGREE)
C
C  Find Lognormal Volume Distribution with same
C   logmean Dp and sigma as THREATS inverted distribution.
C   Begin at 0.01 microns (X=-2.0), end at 1 micron (X=0.),
C    where X is LOG10(Dp) and SG is Geometric Standard Dev.
C   AGREE is fractional agreement (0<AGREE<=1.) between DISTs.
C
      REAL VDIST(37),XD(37),C(37)
      VSUM=0.
      SUM1=0.
      SUM2=0.
      DO 200 K=5,37
         VSUM=VSUM+VDIST(K)
         SUM1=SUM1+VDIST(K)*XD(K)
         SUM2=SUM2+VDIST(K)*XD(K)**2
 200  CONTINUE
C
      XBAR = SUM1 / VSUM
      SIGMA = SQRT ( SUM2/VSUM - XBAR*XBAR )
      VTOT = VSUM / 16.
      ANORM = 1./SQRT(2.*3.141593)*VTOT/SIGMA
      DO 300 K=5,37
         Z = ( XD(K) - XBAR ) / SIGMA
 300     C(K) = ANORM * EXP ( - 0.5 * Z * Z )
      AGREE = 0.
      DO 400 K=5,37
 400     AGREE = AGREE + ABS ( C(K) - VDIST(K) )
      AGREE = 1. - AGREE / VSUM
      SG = 10. ** SIGMA
      DVLOG = 10. ** XBAR
      RETURN
      END
      PROGRAM XOPC
C
C     Transfers OPC data from multiline format created by SAVOPC
C      to single line format used as input to OPCIN.
C     Input and Output files both hold raw OPC data, although
C      Input is counts per channnel and Output is #/cc per channel.
C     XOPC will merge undiluted and diluted OPC data.
C
C     Currently limited to six channels on output (for OPCIN).
C
      CHARACTER*20 IFILE,OFILE
      CHARACTER*72 LINE
      CHARACTER*1 ASK
      LOGICAL NEW
      DIMENSION PN(16),PC(16)
      real meandp(16)
C             Currently set to give volumes (hence
C              masses) of channels 1-7 only.
      real vol(16)
      real sizes(17)
      parameter (pi=3.1415927)
      data sizes /.12,.17,.27,.42,.62,.87,1.17,1.52,9*2./
      NEW=.TRUE.
      TOFF=0.
c             Allows multiple files to be merged
c
      DO 3 i=1,16
        meandp(i)=sqrt(sizes(i)*sizes(i+1))
        vol(i)=pi/6*meandp(i)**3
  3   CONTINUE
c
      MAXCH = 6
 5    WRITE(*,10)
10    FORMAT(/' Enter Input Formatted OPC File Name: ',\)
      READ(*,15) IFILE
15    FORMAT(A20)
      OPEN (11,FILE=IFILE,STATUS='OLD')
      IF (NEW) THEN
        WRITE(*,20)
20      FORMAT(/' Enter Output Formatted OPC File Name: ',\)
        READ(*,15) OFILE
        OPEN (12,FILE=OFILE,STATUS='NEW')
        NEW=.FALSE.
      END IF
      WRITE(*,22)
22    FORMAT(' Enter Dilution Ratio (e.g., 1. or 100.): ',\)
      READ(*,*) DILUT
      WRITE(*,23) TOFF
  23  FORMAT(' Enter Run Starting Time in Hours [',F6.3,']: ',\)
      READ(*,24) DUMMY
  24  FORMAT(F7.3)
      IF (DUMMY.NE.0.) TOFF=DUMMY
      DO 30 I=1,2
        READ(11,25) LINE
25      FORMAT(A)
        WRITE(*,25) LINE
 30   CONTINUE
      N=0
100   READ(11,110,END=500,ERR=400) IH,IM,IS,ISEC
110   FORMAT(I2,1X,I2,1X,I2,4X,I6)
      HTIME=IH+IM/60.+IS/3600.-TOFF
      READ(11,120) (PN(I),I=1,8)
      READ(11,120) (PN(I),I=9,16)
120   FORMAT(8F9.0)
      WRITE(*,130) HTIME,ISEC,(PN(I),I=1,5)
130   FORMAT(1X,F8.3,I8,2X,5F9.0)
      N=N+1
      SEC=FLOAT(ISEC)
      HINT=SEC/3600.
      HTIME=HTIME-HINT/2.
c
c             Find Total Number and Volume Concentrations

      TN=0.
      TV=0.
      DO 140 I=1,16
        PC(I)=DILUT*PN(I)/5./SEC
        TN=TN+PC(I)
 140  CONTINUE
      DO 150 J=1,MAXCH
C  Get volumes in each channel and total.  TV in cu. um per cc
        TV = TV + PC(J)*VOL(J)
 150  CONTINUE
      WRITE(12,350) HTIME,TN,(PC(I),I=1,6)
 350  FORMAT(1X,F7.3,1PE10.3,1X,1P6E10.3)
      GOTO 100
c
c             End of input data
c
400   STOP 'STOPPING ON READ ERROR'
500   CLOSE(11)
      WRITE(*,*) N,' OPC DATA SETS TRANSFERRED'
      WRITE(*,510)
510   FORMAT(' Another OPC Input File [N] ? ',\)
      READ(*,511) ASK
511   FORMAT(A1)
      IF (ASK.EQ.'Y' .OR. ASK.EQ.'y') GOTO 5
      CLOSE(12)
      STOP 'NORMAL COMPLETION'
      END
      PROGRAM HISTOPC
C
C        This program accepts Laser OPC data in #/cc for each channel
C         and generates an ASCII histogram file which may readily be
C         plotted (by ZPLOT) to show the raw OPC size distribution.
C        Six channels are used to generate twelve-line output files.
C        Each output data line has the following form:
C         Diameter (limit, microns) , VDIST (um**3/cc), NDIST (#/cc)
C
      PARAMETER ( PI = 3.141593 )
      PARAMETER ( MAX = 6 , MAX1 = MAX + 1 )
      REAL CNUM(MAX), NDIST(MAX), VDIST(MAX)
      REAL DCUT(MAX1), DAV(MAX), DEL(MAX)
      CHARACTER*30 FNAME,DFILE
C
      DATA DCUT / 0.12, 0.17, 0.27, 0.42, 0.62, 0.87, 1.17 /
C
      DATA SECS,FLOW / 120. , 5. /
C
      DATA FNAME / 'HISTOPC.DAT' /
C
      DO 100 K=1,MAX
         DEL(K) = ALOG10(DCUT(K+1)/DCUT(K))
  100    DAV(K) = SQRT(DCUT(K)*DCUT(K+1))
C
      WRITE(*,110)
  110 FORMAT(/T20,'OPC Simple Histogram Inversion Method'/)
C
      WRITE(*,120) SECS
  120 FORMAT(' Sample Duration [',F6.1,' sec] : ',\)
      READ(*,125) DUMMY
  125 FORMAT(F8.2)
      IF (DUMMY.GT.0.) SECS=DUMMY
      WRITE(*,130) FLOW
  130 FORMAT(' Aerosol Flow Rate [',F5.2,' cc/sec] : ',\)
      READ(*,125) DUMMY
      IF (DUMMY.GT.0.) FLOW=DUMMY
      VOLUME = SECS*FLOW
C
      WRITE(*,140) FNAME
  140 FORMAT(' Enter Output FileName [',A20,'] : ',\)
      READ(*,145) DFILE
  145 FORMAT(A30)
      IF (DFILE.NE.' ') FNAME=DFILE
      OPEN(9,FILE=FNAME,STATUS='NEW')
C
C        Initialize total number & total volume to zero, then sum them.
C
      SUM = 0.
      VOL = 0.
      DO 200 K=1,MAX
         WRITE(*,150) K
  150    FORMAT(' Total Number in Channel ',I2,' : ',\)
         READ(*,160) COUNT
  160    FORMAT(F15.2)
         SUM = SUM + COUNT/VOLUME
         VOL = VOL + COUNT/VOLUME*PI/6.*DAV(K)**3
         NDIST(K) = COUNT/VOLUME/DEL(K)
         VDIST(K) = NDIST(K)*PI/6.*DAV(K)**3
  200 CONTINUE
C
      WRITE(*,250) SUM,VOL
  250 FORMAT(/'  Total Number =',1PE10.3,' /cc',
     #       10X,'Total Volume =',1PE10.3,' um**3/cc'/)
      DO 300 K=1,6
         WRITE(9,275) DCUT(K),VDIST(K),NDIST(K)
         WRITE(9,275) DCUT(K+1),VDIST(K),NDIST(K)
  275    FORMAT(F9.4,1PE12.3,1PE12.3)
  300 CONTINUE
      STOP 'Normal Completion'
      END
```

```
$DEBUG
$LARGE FK,FKS,D,DP,FM
C
      PROGRAM OPCIN
C
C       CINVERSE modified for OPC only, large data input stream.
C       Program Idea by Jim Crump, actualized on the ChemVax, 1980.
C       Modified and cleaned up by Dale Warren to run under
C         Microsoft FORTRAN-77 v3.2 on an IBM AT in March, 1986.
C         Using Single Precision.  Still takes approximately
C          real time (if OPC was on two minutes per sample) to
C          invert the data on the AT!
C
C       Output is unformatted data, mainly dN/dlogDp values DNUM
C         from diameters D1 to D2 at HTIME (TNUM is raw Ntotal)
C         By default, 40 (M1) sections are used from 0.1 to 1.0 um.
C
C          WRITE(22) HTIME,TNUM,D1,D2,(Y(I),I=1,NDP),(DNUM(J),J=1,M1)
C
      LOGICAL AUTO
      CHARACTER*30 IFILE,OFILE,DFILE
      CHARACTER*1 ASK
      COMMON /COMM1/ FK(50,50)
      COMMON /COMM2/ FKS(50,50)
      COMMON /COMM3/ D(50,50)
      COMMON /COMM4/ DP(50,50)
      COMMON /COMM5/ FM(50,50)
      COMMON /CONTRL/ AUTO
      DIMENSION Y(50),YS(50),SIGMA(50),S(50),F(50),V(50)
      DIMENSION DIAM(50),DNUM(50)
C
      DATA ND / 6 /
      DATA IFILE,OFILE / 'DATA.OP ' , 'DATA.OC' /
C
        WRITE(*,10)
  10    FORMAT(' STREAMLINED CINVERSE FOR INVERSION OF OPC DATA SETS '/)
  20    FORMAT(I5)
        WRITE(*,15)
  15    FORMAT(' Automatic OPC Inversion with I/O Files [Y] ? ',\)
        READ(*,16) ASK
  16    FORMAT(A1)
        AUTO = .TRUE.
        IF (ASK.EQ.'N' .OR. ASK.EQ.'n') AUTO = .FALSE.
        IF (AUTO) THEN
           WRITE(*,22) IFILE
  22       FORMAT(' Enter Input OPC Datafile Name [',A20,'] : ',\)
           READ(*,24) DFILE
  24       FORMAT(A30)
           IF (DFILE.NE.' ') IFILE=DFILE
           WRITE(*,26) OFILE
  26       FORMAT(' Enter Output OPC Datafile Name [',A20,'] : ',\)
           READ(*,24) DFILE
```

```
            IF (DFILE.NE.' ') OFILE=DFILE
            OPEN(21,FILE=IFILE,STATUS='OLD')
            OPEN(22,FILE=OFILE,STATUS='NEW',FORM='UNFORMATTED')
            OPEN(20,FILE='OPSTAT.OUT',STATUS='NEW')
         ENDIF
   40    NDP=ND
         NS=0
         WRITE(*,60)
   60    FORMAT(' Enter NUMBER OF QUADRATURE INTERVALS'
     #   ' (EVEN, <50) [40]: ',\)
         READ(*,20) M
         IF (M.EQ.0) M=40
         M1=M+1
         ML1=M-1
         L=2*M+1
         DO 101 I=1,M1
            DO 101 J=1,M1
               D(I,J)=0.
  101          DP(I,J)=0.
         V(1)=1./M/3.
         V(M1)=V(1)
         DO 102 J=2,M,2
  102       V(J)=4./M/3.
         DO 103 J=3,ML1,2
  103       V(J)=2./M/3.
         WRITE(*,61)
   61    FORMAT(' Enter LOWER DIAMETER IN MICRONS [0.10] : ',\)
         READ(*,62) D1
         IF (D1.LE.0.) D1 = 0.10
   62    FORMAT(F16.7)
         WRITE(*,63)
   63    FORMAT(' Enter UPPER DIAMETER IN MICRONS [1.00] : ',\)
         READ(*,62) D2
         IF (D2.LE.0.) D2 = 1.00
C
         R=ALOG10(D2/D1)
         DO 70 J=1,M1
            X=(J-1.)/M
   70       DIAM(J)=D2**X/D1**(X-1.0)
C
C                CALCULATE KERNEL FUNCTIONS
C
         CALL CAL3(NS,M1,D1,D2)
C
         WRITE(*,76)
   76    FORMAT(' Enter ORDER (1 or 2) OF SOBOLEV INVERSION SPACE',
     #   ' [1] : ',\)
         READ(*,20) IFLAG
C
C                CALCULATE DIFFERENCE OPERATOR MATRIX
C
         IF (IFLAG.EQ.2) THEN
            CALL REP2(M)
```

```
       ELSE
          CALL REP1(M)
       ENDIF
C
C                MULTIPLY KERNEL MATRIX BY WEIGHTS FOR QUADRATURE
C
       DO 85 I=1,NDP
          DO 85 J=1,M1
CC               WRITE(*,*) I,J,FK(I,J)
  85         FK(I,J)=FK(I,J)*V(J)
       DO 86 I=1,M1
          DO 86 J=1,M1
             DO 86 K=1,M1
  86            DP(I,J)=DP(I,J)+D(K,I)*V(K)*D(K,J)
C
  80   WRITE(*,120)
 120   FORMAT(' Enter 1 TO ENTER RELATIVE STANDARD',
     # ' DEVIATIONS [Assume equal]: ',\)
       READ(*,20) IZ2
       IF (IZ2.EQ.0) GOTO 150
C
C                Next OPC Data Set if Standard Deviations also Known
C
 100   DO 130 I=1,NDP
          WRITE(*,125) I
 125      FORMAT(' ENTER SIGMA(',I2,') : ',\)
          READ(*,105) SIGMA(I)
 105      FORMAT(E16.7)
 130   CONTINUE
C
C                Next OPC Data Set if Default Standard Deviations Known
C
 150   IF (AUTO) THEN
          READ(21,155,END=900,ERR=900) HTIME,TNUM,(Y(I),I=1,NDP)
 155      FORMAT(1X,F7.3,E10.3,1X,6E10.3)
          WRITE(*,156) HTIME,TNUM,(Y(I),I=1,NDP)
 156      FORMAT(1X,F7.3,1PE10.3,1X,1P6E10.3)
        ELSE
          DO 206 I=1,NDP
             WRITE(*,202) I
 202         FORMAT(' ENTER DATUM Y(',I2,') : ',\)
 206         READ(*,105) Y(I)
       ENDIF
       DO 210 I=1,NDP
          IF (IZ2.EQ.0) THEN
             S(I)=1.
           ELSE
             S(I)=SIGMA(I)**2+0.01*Y(I)**2
             S(I)=SQRT(S(I))
          ENDIF
          YS(I)=Y(I)/S(I)
 210   CONTINUE
       DO 207 I=1,NDP
```

```
            DO 207 J=1,M1
 207           FKS(I,J)=FK(I,J)/S(I)
         IF (NDP.GT.50) STOP 'NDP too large for arrays'
         IF (M1.GT.50) STOP 'M1 too large for arrays'
C
         DO 212 I=1,M1
            DO 212 J=1,M1
               SUM = 0.
               DO 211 LL=1,NDP
                 TERM = FKS(LL,I) * FKS(LL,J)
 211             SUM = SUM + TERM
               FM(I,J) = SUM
CD                FM(I,J)=FM(I,J)+FKS(LL,I)*FKS(LL,J)
 212     CONTINUE
C
C                INVERT DATA
C
         CALL LEMKE(NDP,M1,YS,F,HTIME)
         DO 220 J=1,M1
 220        DNUM(J)=F(J)/R
         IF (AUTO) THEN
            WRITE(22) HTIME,TNUM,D1,D2,(Y(I),I=1,NDP),(DNUM(J),J=1,M1)
          ELSE
            WRITE(12,230)
 230        FORMAT(' SIZE DIST',5X,'DIAM(MICRONS)')
            DO 240 J=1,M1
               WRITE(12,250) DNUM(J),DIAM(J)
 250           FORMAT(1PE11.3,7X,0PF7.3)
 240        CONTINUE
         ENDIF
         WRITE(*,260)
 260     FORMAT(' INPUT DATA',2X,'CALCULATED DATA')
         DO 280 I=1,NDP
            Z=0.0
            DO 270 J=1,M1
 270           Z=Z+FK(I,J)*F(J)
            WRITE(*,290) Y(I),Z
 290        FORMAT(1PE10.3,5X,1PE10.3)
 280     CONTINUE
C
         IF (AUTO) GOTO 150
         WRITE(*,300)
 300     FORMAT(/' More Data (1=New Sigmas, 2=End) [Yes] ? ',\)
         READ(*,20) IX7
         IF (IX7.EQ.0) THEN
            GOTO 150
         ELSEIF (IX7.EQ.1) THEN
            GOTO 100
         ENDIF
 900     STOP
         END
C
C-------------------------------------------------------------
```

```
C
C        CALCULATE ENTRIES OF SECOND DIFFERENCE MATRIX D
C
      SUBROUTINE REP2(M)
C
      COMMON /COMM3/ D(50,50)
      DATA ONE, TWO / 1., 2. /
      H = ONE / M
      DO 10 I=2,M
         D(I,I) = -TWO / H / H
         D(I,I+1)= ONE / H / H
10       D(I,I-1)= ONE / H / H
      RETURN
      END
C
C-----------------------------------------------------------------
C
C        CALCULATE ENTRIES OF FIRST DIFFERENCE MATRIX D
C
      SUBROUTINE REP1(M)
      COMMON /COMM3/ D(50,50)
      DATA ONE / 1.0 /
      H = ONE / M
      DO 10 I = 1,M
         D(I,I)   = -ONE / H
10       D(I,I+1) = ONE / H
      RETURN
      END
C
C-----------------------------------------------------------------
C
        SUBROUTINE LEMKE(NDP,N,Y,F,HTIME)
C
C       QUADRATIC PROGRAMMING ROUTINE LEMKE
C       Note LEMKE's FK is really FKS and its D is really DP !!!
C
      LOGICAL AUTO
      COMMON /COMM2/ FK(50,50)
      COMMON /COMM4/ D(50,50)
      COMMON /COMM5/ FM(50,50)
      COMMON /CONTRL/ AUTO
        COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
      DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),Z(50)
      DIMENSION MBASIS(100),F(50),Y(50),YI(50),SIGMA(50)
      DATA SMOP / 1.E-6 /
        XL1=0.0
   2    FORMAT(I5)
   9    FORMAT(E16.7)
C
      XLO = -99.
      XHI = -99.
      KOUNT = 0
      IF (SMOP.LT.1.E-10) SMOP=1.E-10
```

```
  15  IF (.NOT.AUTO) THEN
        WRITE(*,12) SMOP
  12    FORMAT(' ENTER SMOOTHING PARAMETER [',1PE10.3,'] : ',\)
        READ(*,9) DUMMY
        IF (DUMMY.GT.0.) SMOP=DUMMY
      ENDIF
C
C                        INITIALIZE VALUES
  16    DO 17 I=1,N
           Q(I)=0.
           DO 17 J=1,NDP
  17          Q(I)=Q(I)-FK(J,I)*Y(J)
        DO 13 I=1,N
           DO 13 J=1,N
  13          AM(I,J)=FM(I,J)+NDP*SMOP*D(I,J)
        CALL MATRIX(N)
        CALL INITIA(N)
        IF (IR.EQ.1000) GOTO 40
  14    CALL NEWBAS(N,F)
        IF (IR.EQ.1000) GOTO 20
        CALL SORT(N)
        IF (IR.EQ.1000) GOTO 40
        CALL PIVOT(N)
        GOTO 14
C
C                FIT is Goodness of Fit Index
C       for agreement of Measured Data Y(I) and Inverted Data YI(I)
C       Ideally, FIT=0. using proper autocorrelation to determine
C        smoothing parameter.  In some cases FIT cannot be brought
C        down to zero.
C
  20    FIT=0.
        DO 22 I=1,NDP
           YI(I)=0.
           DO 21 J=1,N
  21          YI(I)=YI(I)+FK(I,J)*F(J)
  22       FIT=FIT+(YI(I)-Y(I))**2/NDP
        FIT=(SQRT(FIT)-1.0)
        KOUNT = KOUNT+1
        WRITE(*,23) SMOP,FIT,KOUNT
  23    FORMAT(/' Smoothing Parameter =',1PE16.7,'  gives FIT=',0PF12.6,
     #        '  on Try #',I3)
C
C       Estimate Better Value for Smoothing Parameter
C
C       HYBRID Stepwise Search / False Position / Secant Method
C       (This is essentially the Secant Method, except we use a
C        reasonable initial guess and use a linear step search to
C        find the sign change, thus bracketing the answer to within
C        an log interval of 1 before starting the secant method.
C        Additionally, we watch the upper and lower bounds so that
C        if the Secant Method ever tries to throw us outside those
C        bounds, we fall back to the slightly lower order false
C         position method.)  Normally stop when FIT < 1.E-5 or
C         when 20 iterations have passed.
C
C
C        Assumes Y is a monotonically increasing function of X
C        Where X (being a log of another variable) is always positive.
C
C        Let XLO <= Xroot <= XHI at all times;     YLO <= 0. <= YHI
C        Let XOLD be prior guess & XNEW latest     (YNEW, YOLD)
C
C        Begin with stepwise search for sign(Y) crossover.
C                XLO = XHI = XOLD = -99. (sign that they are unknown)
C                Guess XNEW, find YNEW.  Sets either LO or HI.
C                Search for other unknown by XNEW=XOLD +/- 1.
C                Iterate till XLO and XHI both have positive values.
C        Update XLO & XHI (YLO, YHI) to keep bounded nicely.
C
         XOLD = XNEW
         YOLD = YNEW
         XNEW = ALOG(SMOP)
         YNEW = FIT
         STEP = 2.*ABS(YNEW)
         IF (STEP.GT.2.) STEP=2.
         IF (STEP.LT.0.05) STEP=0.05
         IF (XHI.LE.-99. .AND. XLO.LE.-99.) THEN
            IF (YNEW.GT.0.) THEN
                XHI = XNEW
                YHI = YNEW
                XX = XNEW - STEP
              ELSE
                XLO = XNEW
                YLO = YNEW
                XX = XNEW + STEP
             ENDIF
          ELSEIF (XHI.LE.-99.) THEN
            IF (YNEW.GE.0.) THEN
                XHI = XNEW
                YHI = YNEW
                XX = FINT(XLO,YLO,XHI,YHI)
             ELSE
                XLO = XNEW
                YLO = YNEW
                XX = XNEW + STEP
           ENDIF
          ELSEIF (XLO.LE.-99.) THEN
            IF (YNEW.LE.O.) THEN
                XLO = XNEW
                YLO = YNEW
                XX = FINT(XLO,YLO,XHI,YHI)
              ELSE
                XHI = XNEW
                YHI = YNEW
                XX = XNEW - STEP
             ENDIF
         ELSE
              IF (YNEW.GT.0.) THEN
                   XHI = XNEW
                   YHI = YNEW
               ELSE
                   XLO = XNEW
                   YLO = YNEW
             ENDIF
             XX = FINT(XOLD,YOLD,XNEW,YNEW)
             IF (XX.LT.XLO .OR. XX.GT.XHI)  XX = FINT(XLO,YLO,XHI,YHI)
         ENDIF
C        IF (Y1.NE.FIT .AND. XL1.NE.0) THEN
C           XL=ALOG(SMOP)-FIT*((XL1-ALOG(SMOP))/(Y1-FIT))
C           XL1=ALOG(SMOP)
C           SMOP=EXP(XL)
C           Y1=FIT
C         ELSEIF (XL1.EQ.0.) THEN
C           XL1=ALOG(SMOP)
C           Y1=FIT
C         ELSE
C           WRITE(*,*) ' Smoothing Parameter Search has converged'
C        ENDIF
C
      SMOP = EXP(XX)
   40 IF (.NOT.AUTO) THEN
        WRITE(*,50) SMOP
   50   FORMAT(' ENTER SMOOTHING PARAMETER (-1. if old OK) ',
     #           '[',1PE10.3,'] : ',\)
        READ(*,9) DUMMY
        IF (DUMMY.GT.0.) SMOP=DUMMY
        IF (DUMMY.GE.0.) GOTO 16
      ENDIF
      IF (SMOP.LT.1.E-15) GOTO 200
      IF (AUTO .AND. ABS(YNEW).GE.1.E-4 .AND.
     #     KOUNT.LT.15 .AND. ABS(XHI-XLO).GE.1.E-4) GOTO 16
  100   WRITE(20,105) KOUNT,EXP(XNEW),YNEW,HTIME
  105   FORMAT(' Iter =',I4,'   Sm.P.=',1PE10.3,'   Fit=',0PF9.6,
     #          '    Hr=',0PF7.3)
        RETURN
C
  200   WRITE(*,*) ' Smoothing Parameter cannot bring Fit Down to 0.'
        RETURN
        END
C
C------------------------------------------------------------------
C
      FUNCTION FINT(X1,Y1,X2,Y2)
C
C               Linear Interpolation to estimate X such that Y=0
C               Used in Secant Method or False Position Searches
C                for Smoothing Parameter to make FIT=0.
C
```

```
      IF (Y1.NE.Y2) THEN
         FINT = X2 - Y2 * (X2-X1)/(Y2-Y1)
       ELSE
         FINT = 0.5*(X1+X2)
      ENDIF
      RETURN
      END
C
C-------------------------------------------------------------------
C
      SUBROUTINE MATRIX(N)
C
C        PURPOSE:  TO INITIALIZE VARIOUS INPUT DATA
C
      COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
      DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),
     #  Z(50),MBASIS(100)
   2     FORMAT(4E16.7)
C
C        IN THE FIRST ITERATION BASIS INVERSE IS AN IDENTITY MATRIX
C
      DO 10 J=1,N
         DO 10 I=1,N
            IF (I.EQ.J) THEN
               B(I,J) = 1.0
            ELSE
               B(I,J) = 0.0
            ENDIF
10    CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------
C
        SUBROUTINE INITIA(N)
C
C       PURPOSE-TO FIND INITIAL ALMOST COMPLEMENTARY SOLUTION
C       BY ADDING ARTIFICIAL VARIABLE Z0
C
        COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
        DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),Z(50),
     #  MBASIS(100)
C
C               SET Z0 EQUAL TO THE MOST NEGATIVE Q(I).
C
        I=1
        J=2
   1    IF (Q(I).GT.Q(J)) I=J
        J=J+1
        IF (J.LE.N) GOTO 1
C
C               UPDATE Q VECTOR
C
```

```
        IR=I
        T1=-Q(IR)
        IF (T1.LE.0.0) GO TO 9
        DO 3 I=1,N
           Q(I)=Q(I)+T1
    3   CONTINUE
        Q(IR)=T1
C
C                 UPDATE BASIS INVERSE AND INDICATOR VECTOR OF
C                 BASIC VARIABLES.
C
        DO 4 J=1,N
           B(J,IR)=-1.0
           W(J)=Q(J)
           Z(J)=0.0
           MBASIS(J)=1
           L=N+J
           MBASIS(L)=J
    4   CONTINUE
        NL1=1
        L=N+IR
        NL2=IR
        MBASIS(IR)=3
        MBASIS(L)=0
        W(IR)=0.0
        Z0=Q(IR)
        L1=1
    7 RETURN
    9   WRITE(*,10)
   10   FORMAT(/5X,'PROBLEM HAS A TRIVIAL COMPLEMENTARY '
     #               'SOLUTION WITH W=Q, Z=0.'/)
        IR=1000
      RETURN
      END
C
C-------------------------------------------------------------
C
        SUBROUTINE NEWBAS(N,F)
C       PURPOSE-TO FIND THE NEW BASIS COLUMN TO ENTER IN TERMS
C       OF THE CURRENT BASIS.
        COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
        DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),Z(50),
     #  MBASIS(100),F(50)
C       IF NL1 IS NEITHER 1 NOR 2 THEN THE VARIABLE Z0 LEAVES
C       THE BASIS INDICATING TERMINATION WITH A COMPLEMENTARY
C       SOLUTION.
C
        IF (NL1.EQ.1) THEN
           NE1=2
           NE2=NL2
C               UPDATE NEW BASIC COLUMN BY MULTIPLYING BY BASIS INVERSE.
           DO 4 I=1,N
              T1=0.0
              DO 3 J=1,N
   3             T1=T1-B(I,J)*AM(J,NE2)
              A(I)=T1
   4        CONTINUE
          ELSEIF (NL1.EQ.2) THEN
            NE1=1
            NE2=NL2
            DO 6 I=1,N
               A(I)=B(I,NE2)
   6        CONTINUE
          ELSE
C           WRITE(*,1)
   1        FORMAT(5X,22HCOMPLEMENTARY SOLUTION)
            CALL PRINT(N,F)
            IR=1000
         ENDIF
         RETURN
         END
C
C-------------------------------------------------------------------
C
         SUBROUTINE SORT(N)
C        PURPOSE-TO FIND PIVOT ROW FOR NEXT ITERATION BY USE OF
C        SIMPLEX MINIMUM RATIO RULE.
         COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
         DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),Z(50),
     #   MBASIS(100)
         I=1
1        IF(A(I).GT.0.0) GO TO 2
         I=I+1
         IF(I.GT.N) GO TO 6
         GO TO 1
2        T1=Q(I)/A(I)
         IR=I
3        I=I+1
         IF(I.GT.N) GO TO 5
         IF(A(I).GT.0.0) GO TO 4
         GO TO 3
4        T2=Q(I)/A(I)
         IF(T2.GE.T1) GO TO 3
         IR=I
         T1=T2
         GO TO 3
5        RETURN
C        FAILURE OF RATIO RULE INDICATES TERMINATION WITH NO
C        COMPLEMENTARY SOLUTION.
6        WRITE(*,7)
7        FORMAT(5X,37HPROBLEM HAS NO COMPLEMENTARY SOLUTION)
         WRITE(*,8) L1
8        FORMAT(10X,13HITERATION NO.,I4)
         IR=1000
         RETURN
         END
```

```
C
C--------------------------------------------------------------------
C
        SUBROUTINE PIVOT(N)
C       PURPOSE-TO PERFORM THE PIVOT OPERATION BY UPDATING
C       THE INVERSE OF THE BASIS AND Q VECTOR.
        COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
        DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),Z(50),
     #  MBASIS(100)
        DO 1 I=1,N
1       B(IR,I)=B(IR,I)/A(IR)
        Q(IR)=Q(IR)/A(IR)
        DO 3 I=1,N
        IF(I.EQ.IR) GO TO 3
        Q(I)=Q(I)-Q(IR)*A(I)
        DO 2 J=1,N
        B(I,J)=B(I,J)-B(IR,J)*A(I)
2       CONTINUE
3       CONTINUE
C       UPDATE THE INDICATOR VECTOR OF BASIC VARIABLES.
        NL1=MBASIS(IR)
        L=N+IR
        NL2=MBASIS(L)
        MBASIS(IR)=NE1
        MBASIS(L)=NE2
        L1=L1+1
        RETURN
        END
C
C--------------------------------------------------------------------
C
        SUBROUTINE PRINT(N,F)
C       PURPOSE-TO PRINT SOLUTION TO COMPLEMENTARY PROBLEM AND
C       ITERATION NUMBER.
        COMMON AM,Q,L1,B,NL1,NL2,A,NE1,NE2,IR,MBASIS,W,Z
        DIMENSION AM(50,50),Q(50),B(50,50),A(50),W(50),Z(50),
     #            MBASIS(100),F(50)
CC      WRITE(*,1) L1
   1    FORMAT(10X,'ITERATION # ',I4)
        DO 20 I=1,N
           DO 10 J=1,N
              IS=J
              IF (MBASIS(J).EQ.1) GO TO 10
              IF (MBASIS(J+N).EQ.I) GO TO 15
  10       CONTINUE
           QS=0.0
           F(I)=QS
           GOTO 20
  15       CONTINUE
           IF (Q(IS).GE.0.) GOTO 16
           Q(IS)=0.0
  16       F(I)=Q(IS)
  20    CONTINUE
```

```
          RETURN
          END
C
C-----------------------------------------------------------------
C
          SUBROUTINE CAL3(NS,M1,D1,D2)
C         DOUBLE PRECISION FK,D1,D2,E
          LOGICAL AFLAG
       COMMON /COMM1/ FK(50,50)
          DIMENSION E(6,41),DIAM(41)
C
C         OPC Response Matrix is E.  It is too large to enter directly
C          in DATA because Continuations > 9 unacceptable to MICROSOFT.
C         Break up the E Matrix so it can be read in by DATA statements.
C         EQUIVALENCE with these smaller matrices did not work.
C         Hence assign E to sub-E matrices once, when AFLAG .TRUE.
C
       DIMENSION E0(6,10),E1(6,10),E2(6,10),E3(6,11)
C      EQUIVALENCE ( E(1,1)  , E0(1,1) )
C      EQUIVALENCE ( E(1,11) , E1(1,1) )
C      EQUIVALENCE ( E(1,21) , E2(1,1) )
C      EQUIVALENCE ( E(1,31) , E3(1,1) )
       DATA E0 / 0.0630, 0.0100, 0.0000, 0.0000, 0.0000, 0.0000,
     1  0.0811, 0.0130, 0.0008, 0.0000, 0.0000, 0.0000,
     2  0.0977, 0.0176, 0.0011, 0.0000, 0.0000, 0.0000,
     3  0.1115, 0.0238, 0.0009, 0.0000, 0.0000, 0.0000,
     4  0.1263, 0.0252, 0.0015, 0.0000, 0.0000, 0.0000,
     5  0.1419, 0.0607, 0.0024, 0.0000, 0.0000, 0.0000,
     6  0.1568, 0.1252, 0.0029, 0.0000, 0.0000, 0.0000,
     7  0.1699, 0.2324, 0.0029, 0.0000, 0.0000, 0.0000,
     8  0.1816, 0.3533, 0.0026, 0.0000, 0.0000, 0.0000,
     9  0.1896, 0.4864, 0.0020, 0.0000, 0.0000, 0.0000 /
       DATA E1 / 0.1954, 0.6181, 0.0014, 0.0000, 0.0000, 0.0000,
     1  0.1959, 0.7315, 0.0008, 0.0000, 0.0000, 0.0000,
     2  0.1933, 0.8282, 0.0002, 0.0000, 0.0000, 0.0000,
     3  0.1837, 0.8763, 0.0000, 0.0000, 0.0000, 0.0000,
     4  0.1701, 0.8924, 0.0000, 0.0000, 0.0000, 0.0000,
     5  0.1479, 0.8295, 0.0039, 0.0000, 0.0000, 0.0000,
     6  0.1215, 0.7252, 0.0847, 0.0003, 0.0000, 0.0000,
     7  0.0921, 0.5738, 0.2632, 0.0008, 0.0000, 0.0000,
     8  0.0638, 0.4166, 0.4627, 0.0013, 0.0000, 0.0000,
     9  0.0425, 0.2833, 0.6479, 0.0013, 0.0000, 0.0000 /
       DATA E2 / 0.0275, 0.1798, 0.7981, 0.0009, 0.0000, 0.0000,
     1  0.0268, 0.1523, 0.8375, 0.0000, 0.0000, 0.0000,
     2  0.0297, 0.1419, 0.8339, 0.0000, 0.0000, 0.0000,
     3  0.0326, 0.1241, 0.7693, 0.0281, 0.0000, 0.0000,
     4  0.0350, 0.1023, 0.6839, 0.0981, 0.0005, 0.0003,
     5  0.0364, 0.0764, 0.5752, 0.2265, 0.0015, 0.0009,
     6  0.0371, 0.0505, 0.4641, 0.3713, 0.0025, 0.0015,
     7  0.0364, 0.0288, 0.3668, 0.5236, 0.0030, 0.0019,
     8  0.0349, 0.0113, 0.2857, 0.6669, 0.0030, 0.0020,
     9  0.0318, 0.0058, 0.2533, 0.7663, 0.0015, 0.0013 /
       DATA E3 / 0.0289, 0.0045, 0.2305, 0.8277, 0.0000, 0.0005,
     1  0.0287, 0.0076, 0.1968, 0.7808, 0.0000, 0.0000,
     2  0.0309, 0.0109, 0.1428, 0.6635, 0.0477, 0.0000,
     3  0.0345, 0.0137, 0.0739, 0.4761, 0.3312, 0.0031,
     4  0.0366, 0.0159, 0.0143, 0.2962, 0.6245, 0.0076,
     5  0.0316, 0.0163, 0.0071, 0.2167, 0.7208, 0.0151,
     6  0.0250, 0.0160, 0.0171, 0.1757, 0.7250, 0.0382,
     7  0.0224, 0.0156, 0.0200, 0.1386, 0.6635, 0.1327,
     8  0.0214, 0.0151, 0.0203, 0.1030, 0.5773, 0.2540,
     9  0.0224, 0.0146, 0.0171, 0.0692, 0.4583, 0.4110,
     1  0.0240, 0.0140, 0.0130, 0.0360, 0.3310, 0.5770 /
      DATA DIAM /
     1  0.1000,0.1059,0.1122,0.1189,0.1259,0.1334,0.1413,
     2  0.1496,0.1585,0.1679,0.1778,0.1884,0.1995,0.2113,
     3  0.2239,0.2371,0.2512,0.2661,0.2818,0.2985,0.3162,
     4  0.3350,0.3548,0.3758,0.3981,0.4217,0.4467,0.4732,
     5  0.5012,0.5309,0.5623,0.5957,0.6310,0.6683,0.7079,
     6  0.7499,0.7943,0.8414,0.8913,0.9441,1.000 /
        DATA AFLAG /.TRUE./
C
      IF (D2.GT.1.011) THEN
         WRITE(*,15)
15       FORMAT(' UPPER DIAMETER IS TOO LARGE;'/
     #          ' OPC CALIBRATION GOES UP TO 1 MICRON.')
         RETURN
      ENDIF
      IF (AFLAG) THEN
        DO 777 J=1,41
           DO 776 I=1,6
              IF (J.LE.10) THEN
                 E(I,J)=E0(I,J)
               ELSEIF (J.LE.20) THEN
                 E(I,J)=E1(I,J-10)
               ELSEIF (J.LE.30) THEN
                 E(I,J)=E2(I,J-20)
               ELSEIF (J.LE.41) THEN
                 E(I,J)=E3(I,J-30)
               ELSE
                 STOP 'EAA Array Out of Range'
              ENDIF
776        CONTINUE
777        WRITE(*,778) J,(E(I,J),I=1,6)
778           FORMAT(' E(I,',I2,') = ',6F7.4)
        AFLAG=.FALSE.
      ENDIF
      DO 40 J=1,M1
         X=(J-1.)/(M1-1.)
         D=D2**X/D1**(X-1.)
         IF (D.LT.0.1) THEN
            DO 26 I=1,6
26             FK(I+NS,J)=0.0
          ELSE
            DO 35 K=1,41
               KP=K-1
              IF (D.LT.DIAM(K)) GOTO 36
35         CONTINUE
36         DO 37 I=1,6
              FK(I+NS,J)=E(I,KP)+ALOG(D/DIAM(KP))*
   #            (E(I,KP+1)-E(I,KP))/ALOG(DIAM(KP+1)/DIAM(KP))
37         CONTINUE
        ENDIF
40   CONTINUE
     NS=NS+6
     RETURN
     END
```