# The Physics of Granular Systems

by

**Gary Michael Gutt**

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California, U.S.A.

1989

(Submitted 10 May 1989)

## Acknowledgements

It is often said that the value of an excellent advisor in graduate school is beyond measure. I have been fortunate to have had two such advisors in Peter Haff and Tom Tombrello. Peter's enthusiasm for granular studies is what led me into these investigations. For his support and for his critiques of this thesis and my other papers I am truly grateful. Tom's interest in my work and advice are also greatly appreciated.

For all of their help and comradeship since the beginning of my adventure at Caltech, I am indebted to the fellow members of the physics graduate class which entered in 1982, with special appreciation to Jim Kaufman for these many years of friendship. For many interesting discussions (scientific and otherwise), I would like to thank the "sandbag" subgroup of the Brown Bag group (Bob and Suzanne Anderson and Brad Werner), Tom Drake, and the other members of the Brown Bag group. For providing and maintaining the advanced computing resources which were so important in my work, I am grateful to Geoffrey Fox, Heidi Lorenz-Wirzba, Jon Flower, Adam Kolawa and the other members of the Caltech Concurrent Computation Project.

My work has been supported in part by grants from the Army Research Office [DAAL03-86-K-0123] and the Department of Energy [DE-FG22-86-PC90959], and by a Shell Graduate Fellowship.

Finally, for making it all possible, and for their love and encouragement, I dedicate this thesis to my parents.

## Abstract

To gain an improved understanding of the physics of granular systems, investigations of two aspects of continuum theories of granular flows and a new paradigm for modelling granular systems are presented.

The analogy between a flowing granular system and a molecular gas allows the adaptation of elements of the kinetic theory of gases to a continuum theory of granular flow. Two significant areas of difference between gases and granular materials are the additional degrees of freedom due to the spin and surface roughness of the grains and the boundary conditions between a granular flow and a bounding surface. Using techniques from the field of nonequilibrium thermodynamics, equations of motion and constitutive relations are obtained which include the effects of the spin and surface roughness of the grains. Also, boundary conditions on continuum theories for flows of smooth grains are presented, emphasizing the need to allow for several "slip" degrees of freedom.

In order to obtain a more general description of large scale effects in granular systems involving both flowing and static assemblies of grains, a model is developed (the lattice grain dynamics paradigm) for simulating systems containing large numbers of grains. This model is based on concepts from the fields of cellular automata, lattice gases, and particle dynamics simulations.

## Table of Contents

## Table of Contents (continued)

# Introduction

In discussing the physics of granular systems, the most obvious first question is: "What is a granular system?" For the purposes of this study, a granular system will be defined as consisting of a large number (greater than 100) of macroscopic grains (linear dimensions greater than 0.1 millimeters) of solid matter, with particle number densities ranging from compacted (as in a pile of sand grains in a gravitational field) to disperse (as in particles in a planetary ring). It will be assumed throughout that the grains obey the laws of classical physics, and thus relativistic and quantum mechanical effects can be ignored.

The above definition encompasses a wide range of common physical phenomena: Granular systems are observed in geophysical settings as rock slides, sand dunes, sand storms, sediments, and snow avalanches. In industry, they are found in connection with the processing of cereal grains, coal, gravel, and oil shale. And in astrophysics, they are seen in the form of planetary rings.

In this thesis, we will not concern ourselves with entirely static granular systems, which can be treated using the science of soil mechanics. Instead we will treat problems in which all or at least a significant portion of the grains are in motion relative to the boundaries and to each other. In order to keep our investigations down to a manageable size, we will only consider systems of uniformly-sized, spherical grains; the effects of a distribution in grain shapes and sizes are beyond the scope of the present study.

A granular system may exhibit many of the same characteristics as a fluid (*e.g.,* it may have a self-bounding free surface in a gravitational field and it generally will conform to the shape of a bounding wall on length scales larger than a grain diameter) and thus may be modelled using techniques borrowed from continuum theories of fluid mechanics. However, these systems also differ from fluids in several significant ways: on a macroscopic scale, a compacted granular system can support a shear stress even in the absence of a shearing velocity. Thus, the free surface of a pile of grains need not be perpendicular to the local direction of gravitational acceleration. On a microscopic scale, the collisions and sliding contacts

in a granular system will dissipate the kinetic energy of the particles, whereas the collisions between the molecules of a gas will be perfectly elastic. This means that the relative motions in a collection of grains will be damped out over time, and that maintaining a constant level of activity among the grains will require a constant input of energy from an external source. Also, the description of a granular system as a continuum may break down in those instances in which the motions of a few individual grains are of significance (*e.g.,* the free surface of a granular flow down an inclined slope may not be well defined due to the presence of saltating grains (grains which are bouncing along the surface)).

One of the earliest attempts to formulate a continuum theory for fluidized granular systems was presented by Bagnold (1954) and was based upon a consideration of the microscopic interactions between grains. His theoretical arguments and experimental observations advanced the idea that a cohesionless granular system (with negligible interstitial fluid effects) subjected to a uniform shearing motion would exhibit a shear stress proportional to the square of the rate of shear (similar to the behavior of a fluid in the turbulent regime). He was able to apply this theory to the flow of sand down an inclined slope, but with only moderate success.

A theory based on continuum mechanics and thermodynamics was given by Goodman and Cowin (1972) and extended in Savage (1979). Here, the volume fraction of space filled by the granular material was treated as an explicit variable, and equations governing its evolution in time were postulated. The possibility of spin and surface roughness for the particles was not considered. The coefficients in the constitutive equations were not derived from any consideration of particle-particle interactions, but were instead given *ad hoc* values so as to obtain the desired functional form for the dissipative part of the stress tensor. In addition, the physical interpretations of some of the quantities introduced in the equations governing the volume fraction are unclear. This theory and ones similar to it were used by several investigators (Savage (1979), Nunziato *et al.* (1980), and Passman *et al.* (1980)) to solve simple problems in the flow of granular materials down chutes. The boundary conditions used in solving these problems included a no-slip condition on the

average flow velocity and an arbitrarily chosen value of the volume fraction at the boundaries.

The Goodman-Cowin theory does not take into account the fact that a granular flow is made up of independent particles and that the velocities of these particles will fluctuate about the average flow velocity. These velocity fluctuations were first incorporated into continuum theories of granular flows by Kanatani (1979) and Ogawa *et al.* (1980). A complete, self-consistent continuum theory of granular flows including the concept of fluctuation velocity was derived by Haff (1983) using a heuristic approach to the development of the constitutive relations. This approach facilitated the physical interpretation of the terms in the equations of motion and the constitutive relations and displayed the connection between the microscopic parameters (*e.g.,* the coefficient of restitution in a collision) and the macroscopic effects (*e.g.,* plug formation in Couette flow).

The similarity between a flowing granular system and the kinetic theory model of a gas has led several investigators to attempt a derivation from first principles of the equations of motion and the constitutive relations for granular systems, using techniques from the dense gas theory of Chapman and Enskog. Here, the grains are assumed to be sufficiently agitated that they do not have enduring contacts, but instead undergo only momentary, binary collisions. A Maxwellian distribution for grain velocities is usually assumed; and the equations of motion and constitutive relations are obtained by integrating the results of these binary collisions over this velocity distribution. Savage and Jeffery (1981), Jenkins and Savage (1983), and Lun *et al.* (1984) have used this technique for systems of identical, smooth, inelastic spheres. Lun and Savage (1987) have extended the theory to include rough, spinning spheres. The difficulty with this approach is that the complexity of the integrations renders them impossible to evaluate for all but the simplest possible problems. In addition, the resulting constitutive relations contain terms whose physical interpretation is difficult to ascertain.

The question of boundary conditions to be used for these continuum theories has received only limited attention. Most investigators have simply assumed a no-slip condition on the average flow velocity at a boundary and have made various

arbitrary choices for the boundary conditions on other variables (*e.g.*, the fluctuation velocity may be set to zero at a boundary). Two attempts have been made at deriving the boundary conditions on a granular flow. Hui *et al.* (1984) derived a set of boundary conditions using the heuristic approach of Haff (1983). Jenkins and Richman (1986) used the Chapman-Enskog techniques to obtain their version of the boundary conditions. The problems associated with both of these sets of boundary conditions are discussed in detail in Chapter 2.

The advent of powerful, high speed (and economical) computers has led to the idea of modelling granular systems by simply calculating the motions of the individual grains. Efforts along these lines (detailed in Chapter 3) have been limited to relatively few particles by the complexity of these calculations. The recent surge of interest in cellular automata has provided the impetus for inventing a new technique for modelling granular systems with many particles that does not involve continuum theories and differential equations.

In this thesis, two aspects of the continuum theories of granular flows and a new approach to the modelling of granular systems in general (using cellular automata instead of differential equations) will be presented. In Chapter 1, a continuum theory of granular flows which includes the effects of surface roughness and spin of the grains is derived using the heuristic approach of Haff (1983) and using techniques from continuum mechanics and nonequilibrium thermodynamics. In Chapter 2, the elusive problem of boundary conditions on flows of smooth, nonspinning grains is treated; and it is shown that the simple boundary conditions of fluid mechanics cannot, in general, be used for granular flows. Finally, in Chapter 3, the lattice grain dynamics paradigm for modelling granular systems is presented.

# Chapter 1. A Continuum Theory of Granular Flow Including Spin

The theoretical study of highly agitated granular flows has been significantly advanced in recent years by the application of ideas from the kinetic theory of gases. Intuitively, the random motions and collisions of grains in an agitated flow are suggestive of the motions and collisions of molecules in a gas. By making the analogy between the fluctuation velocity of the grains and the thermal velocity of molecules in a gas, most of the concepts (pressure, viscosity, and thermal conductivity) and equations of motion of the kinetic theory can be carried over to the theory of granular flow. There are, however, several significant differences between the collisions of grains and the collisions of molecules that must be taken into account when making this analogy. These include the inelasticity of the grain collisions and subsequent loss of kinetic energy into heat, the spin and surface roughness of the grains, the higher number density of particles per unit volume and the short mean free path relative to a grain diameter for granular systems, and the differences in the treatment of the boundary conditions.

The first of these effects has been incorporated by careful treatment of the fluctuation of individual grain velocities about the average flow velocity. The magnitude of these fluctuations (the "fluctuation velocity," or, in analogy with the kinetic theory of gases, the "thermal velocity") is important because it largely determines the relative velocity of grains in a collision. Since energy and momentum in a granular flow (as in a dense gas) are transmitted predominantly by collisional transport (rather than kinetic transport), this thermal velocity will play a significant role in the constitutive relations for the pressure and shear stress. Thus, an equation (the "thermal energy equation") governing the generation, diffusion, and loss of the thermal velocity is a necessary component of a theory of granular flow. The inelasticity of grain-grain collisions is incorporated into the theory as a loss term in this equation. The first attempt to include the thermal velocity in a theory of granular flow was made by Ogawa *et al.* (1980). When the techniques of statistical mechanics are used to derive the equations of motion and the constitutive relations for a granular

flow (Savage and Jeffrey (1981), Jenkins and Savage (1983), Lun *et al.* (1984), and Shen and Ackermann (1984)), the thermal velocity and its governing equation are a result, just as in the kinetic theory of gases. A more physically intuitive approach to including these effects has been taken by Haff (1983).

All of these theories have assumed that the grains are smooth, identical spheres and have ignored any possible effects of the spin of the grains. If we consider the example of grains confined between parallel plates and undergoing shearing motion (Couette flow), we can see that the possible spin of the grains may be important in determining the shear stress, inasmuch as they may act as "ball bearings" and influence the effective viscosity of the system. Thus, it would seem worthwhile to include spin in the theory of granular flow. One approach to the incorporation of spin and surface roughness effects was presented by Kanatani (1979), in which equations of motion for the average translational and spin velocities and constitutive relations based on the cell model were derived. However, the fluctuation in spin velocity of the grains (the "spin thermal velocity") was neglected. Another approach has been given by Lun and Savage (1986), once again using the techniques of statistical mechanics. However, the mathematical complexity of this method only allowed for the solution of the problem of steady state Couette flow for the special case of uniform translational and spin thermal velocities. The possibility of thermal energy diffusion was not considered in this special case.

In this chapter, methods from the theory of nonequilibrium thermodynamics are utilized along with the intuitive approach of Haff (1983) to obtain the equations of motion and the constitutive relations, including the effects of diffusion of translational and spin thermal velocities. This formalism is then used to obtain solutions to various problems, with the goal of gaining an insight into the physical results of including the effects of spin and surface roughness into the theory.

## The Equations of Motion

In this formalism, the concept of thermal velocity as introduced by Ogawa *et al.* (1980) will be retained. Thus, to include the effects of spin into the theory of

granular flow, we must allow four new degrees of freedom: the three components of the average spin angular velocity ($\vec{\omega}$), and the r.m.s. magnitude of the spin fluctuation angular velocity ($W$). This requires the addition of four new equations of motion for the spin variables and the modification of the existing equations to allow for the exchange of angular momentum and energy between translational and spin degrees of freedom. The energy flow relationships of this model are shown in figure 1.1. It should be noted that both the average spin velocity and the spin fluctuation velocity represent reservoirs which can absorb energy from or supply energy to the translational degrees of freedom.

*The continuity equation*

The conservation of mass gives the usual continuity equation:

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x_j}(\rho u_j), \tag{1.1}$$

where $\rho$ is the bulk density of the granular flow, and $u_j$ is the average flow velocity.

Since most granular systems of interest involve a high density of grains (*i.e.*, the volume fraction occupied by the grains is typically $\sim .5$), the theory given here assumes an incompressible flow: $\rho =$ constant and

$$\vec{\nabla} \cdot \vec{u} = 0.$$

*The momentum equations*

The derivation of the translational and spin momentum equations presented here follows that given in de Groot and Mazur (1962). Since this derivation depends upon the second law of thermodynamics and the use of the concept of entropy, the question arises as to whether these principles apply to granular flows. Inasmuch as a granular flow is a dynamical system with some probabilistic evolution, it seems reasonable that the concept of entropy can be applied to this case. Because entropy gives a measure of the uncertainty in the state of a system, and the uncertainty in the state of an agitated granular flow increases with time, the second law of thermodynamics should apply here (Oshima (1978)).

From the conservation of linear and angular momentum, the evolution of linear and spin momentum can be expressed in terms of the pressure tensor:

$$\rho \frac{du_i}{dt} = \rho g_i - \frac{\partial}{\partial x_j} P_{ij}, \tag{1.2}$$

$$\rho \theta \frac{d\omega_i}{dt} = \epsilon_{ijk} P_{jk}, \tag{1.3}$$

where $P_{ij}$ is the pressure tensor, $g_i$ is the gravitational acceleration, $\theta$ is the moment of inertia per unit mass of a grain, and $\omega_i$ is the average spin angular velocity. Equation 1.2 is the familiar expression from fluid mechanics relating the rate of change in average flow velocity to the gravitational acceleration and the gradient of the pressure. Equation 1.3 expresses the fact that changes in the average spin velocity are driven by the antisymmetric portion of the pressure tensor.

To obtain the dependence of the pressure tensor on the average flow velocity and average spin velocity, the usual techniques from the theory of nonequilibrium thermodynamics will be used. We begin with the expression for the rate of entropy production per unit volume (de Groot and Mazur (1962) equation XII.23):

$$\sigma_e = -\vec{J}_q \cdot \frac{\vec{\nabla} T}{T^2} - p \frac{\vec{\nabla} \cdot \vec{u}}{T} - \frac{\sqcap^s : (\vec{\nabla}\vec{u})^s}{T} - \vec{\sqcap}^a \cdot \frac{(\vec{\nabla} \times \vec{u} - 2\vec{\omega})}{T} \geq 0,$$

where $J_q$ is the flux of "thermal" energy, $T$ is the "temperature" of the grains, $p$ is one third the trace of the pressure tensor, $\sqcap^s$ is the symmetric part of the pressure tensor with zero trace, $(\vec{\nabla}\vec{u})^s$ is the symmetric part of the gradient of the average flow velocity (with zero trace), $\sqcap^s : (\vec{\nabla}\vec{u})^s = \sqcap^s_{ik} (\vec{\nabla}\vec{u})^s_{ki}$, and $\vec{\sqcap}^a$ is the axial vector corresponding to the antisymmetric part of the pressure tensor. (Thus $P_{ij} = p\delta_{ij} + \sqcap^s_{ij} + \epsilon_{ijk} \sqcap^a_k$ .)

Since the flow is assumed to be incompressible, the term involving $\vec{\nabla} \cdot \vec{u}$ is zero. Note that this expression consists of the sum of products of thermodynamic forces and fluxes. Following standard practice in nonequilibrium thermodynamics for obtaining the first order constitutive relations, we will now assume that these fluxes are linearly related to the thermodynamic forces and that the fluid properties have isotropic symmetry, and obtain the following set of phenomenological equations:

$$\sqcap^s = -2\eta_t (\vec{\nabla}\vec{u})^s \quad \text{or} \quad \sqcap^s_{ij} = -\eta_t \left( \frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right), \tag{1.4}$$

$$\vec{\Pi}^a = -\eta_r(\vec{\nabla} \times \vec{u} - 2\vec{\omega}) \quad \text{or} \quad \Pi_i^a = -\eta_r \left( \epsilon_{ilm} \frac{\partial u_m}{\partial x_l} - 2\omega_i \right), \tag{1.5}$$

where $\eta_t$ is the translational coefficient of viscosity, and $\eta_r$ is the rotational coefficient of viscosity. (Rotational viscosity is an effect in which a difference between the average rotational motion of the flow ($\frac{1}{2}\vec{\nabla} \times \vec{u}$) and the average spin of the grains ($\omega$) will generate an asymmetry in the pressure tensor.)

Inserting these relations into the equations of motion (equations 1.2 and 1.3) and substituting for the convective derivative gives

$$\frac{\partial}{\partial t}(\rho u_i) = \rho g_i$$

$$-\frac{\partial}{\partial x_j}\left[ p\delta_{ij} + \rho u_i u_j - \eta_t \left( \frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right) - \epsilon_{ijk}\eta_r \left( 2\omega_k - \epsilon_{klm}\frac{\partial u_m}{\partial x_l} \right) \right], \tag{1.6}$$

$$\frac{\partial}{\partial t}(\rho\theta\omega_i) = -\frac{\partial}{\partial x_j}(\rho\theta\omega_i u_j) - 2\eta_r \left( 2\omega_i - \epsilon_{ilm}\frac{\partial u_m}{\partial x_l} \right). \tag{1.7}$$

*The energy equations*

The energy equations describe the generation, flow, transfer, and loss of translational and spin thermal energies.

The derivation of the translational thermal energy equation begins with the total translational energy equation:

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\rho u^2 + \frac{1}{2}\rho v^2\right) = -\frac{\partial}{\partial x_j}\left[ u_i P_{ij} + \frac{1}{2}\rho(u^2 + v^2)u_j \right] + \rho u_i g_i - \frac{\partial}{\partial x_j} Q_{(t)j} - I_t, \tag{1.8}$$

where $v$ is the translational fluctuation velocity, $\frac{1}{2}\rho u^2$ is the kinetic energy associated with the average flow velocity, $\frac{1}{2}\rho v^2$ is the kinetic energy associated with the thermal velocity, $Q_{(t)j}$ is the flux of translational thermal energy, and $I_t$ includes the losses due to inelasticity and friction and the transfer of energy to or from spin thermal energy.

Multiplying the linear momentum equation (equation 1.2) by $u_i$ and subtracting from equation 1.8 leaves

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\rho v^2\right) = -P_{ij}\frac{\partial u_i}{\partial x_j} - \frac{\partial}{\partial x_j}\left(\frac{1}{2}\rho v^2 u_j\right) - \frac{\partial}{\partial x_j} Q_{(t)j} - I_t.$$

Substituting for the thermal flux and the pressure tensor,

$$Q_{(t)j} = -K_t \frac{\partial}{\partial x_j}(\frac{1}{2}\rho v^2),$$

$$P_{ij} = p\delta_{ij} - \eta_t \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) - \epsilon_{ijk}\eta_r \left(2\omega_k - \epsilon_{klm}\frac{\partial u_m}{\partial x_l}\right),$$

gives the final form of the translational thermal energy equation:

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho v^2) = \left[\eta_t \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) + \epsilon_{ijk}\eta_r \left(2\omega_k - \epsilon_{klm}\frac{\partial u_m}{\partial x_l}\right)\right]\frac{\partial u_i}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\frac{1}{2}\rho v^2 u_j) - K_t \frac{\partial}{\partial x_j}(\frac{1}{2}\rho v^2)\right] - I_t, \quad (1.9)$$

where $K_t$ is the translational thermal diffusion coefficient.

The evolution of the total spin kinetic energy is given by

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho\theta\omega^2 + \frac{1}{2}\rho\theta W^2) = -\frac{\partial}{\partial x_j}[\frac{1}{2}\rho\theta(\omega^2 + W^2)u_j] - \frac{\partial}{\partial x_j}Q_{(r)j} - I_r, \quad (1.10)$$

where $W$ is the spin fluctuation velocity (the analog for spin of the translational thermal velocity), $Q_{(r)j}$ is the flux of spin thermal energy, and $I_r$ includes the losses due to friction and the transfer of energy to or from translational thermal energy. Multiplying the equation for spin angular momentum (equation 1.3) by $\omega_i$ and subtracting from equation 1.10:

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho\theta W^2) = -\omega_i\epsilon_{ijk}P_{jk} - \frac{\partial}{\partial x_j}(\frac{1}{2}\rho\theta W^2 u_j) - \frac{\partial}{\partial x_j}Q_{(r)j} - I_r.$$

Substituting for the thermal flux and the pressure tensor,

$$Q_{(r)j} = -K_r \frac{\partial}{\partial x_j}(\frac{1}{2}\rho\theta W^2),$$

$$\epsilon_{ijk}P_{jk} = -2\eta_r \left(2\omega_i - \epsilon_{ilm}\frac{\partial u_m}{\partial x_l}\right),$$

gives the spin thermal energy equation:

$$\frac{\partial}{\partial t}(\frac{1}{2}\rho\theta W^2) = 2\eta_r\omega_i \left(2\omega_i - \epsilon_{ilm}\frac{\partial u_m}{\partial x_l}\right) - \frac{\partial}{\partial x_j}\left[(\frac{1}{2}\rho\theta W^2 u_j) - K_r \frac{\partial}{\partial x_j}(\frac{1}{2}\rho\theta W^2)\right] - I_r, \quad (1.11)$$

where $K_r$ is the rotational thermal diffusion coefficient.

It is interesting to note that the first term on the right hand side of the spin thermal energy equation may be positive or negative depending upon the average spin and flow velocities. That this term may lead to a "cooling" of the spin thermal velocity under proper conditions is analogous to the cooling experienced by an expanding gas.

## Constitutive Relations

In order to obtain the constitutive relations, we will assume that our granular system is similar to a dense gas. In a dense gas, the mean free path of a molecule is no longer much greater than the diameter of a molecule; consequently, collisional transport of linear and angular momentum dominates over transport by free particle diffusion. In order to evaluate collisional transport in a dense gas, we make use of the cell model, in which each particle is regarded as bouncing around in a cell defined by the nearest neighbor particles. The inclusion of spin effects requires a knowledge of the results of the collision of two spinning particles. To calculate these results, we employ a model proposed by Lun and Savage (1986).

*The particle collision model*

The grain particles are assumed to be rough, rigid spheres of mass $m$ and diameter $d$ (figure 1.2). In this model, the components of the relative surface velocity ($\vec{g}$) after the collision are related to the components before the collision by the normal and tangential coefficients of restitution:

$$g'_{\perp} = -e \;\; g_{\perp},$$

$$g'_{\parallel} = -\beta \;\; g_{\parallel}.$$

Using the conservation laws of linear and angular momentum, it can be shown that the changes in translational velocities and spins are given by

$$\vec{c}'_2 - \vec{c}_2 = n_2\vec{c}_{12} + (n_1 - n_2)(\vec{k}\cdot\vec{c}_{12})\vec{k} - n_2 d(\vec{k}\times\vec{\Omega}), \tag{1.12}$$

$$\vec{\omega}'_2 - \vec{\omega}_2 = -\frac{mdn_2}{2I}[(\vec{k}\times\vec{c}_{12}) - d(\vec{k}\cdot\vec{\Omega})\vec{k} + d\vec{\Omega}], \tag{1.13}$$

where $\vec{c}_1$ is the translational velocity of particle 1; $\vec{c}_2$ is the translational velocity of particle 2; $\vec{c}_{12} = \vec{c}_1 - \vec{c}_2$; $\vec{k}$ is a unit vector from the center of particle 1 to the center of particle 2; $n_1 = \frac{(1+e)}{2}$; $n_2 = \frac{(1+\beta)}{2}\frac{K}{(1+K)}$; $K = \frac{4I}{md^2} = \frac{2}{5}$ for a solid, homogeneous sphere; $I$ = moment of inertia of a particle about its center of mass; $\vec{\omega}_1$ is the angular velocity of particle 1; $\vec{\omega}_2$ is the angular velocity of particle 2; and $\vec{\Omega} = (\vec{\omega}_1 + \vec{\omega}_2)/2$.

If we subtract the change in translational kinetic energy (T.K.E.) of particle 1 from the change in T.K.E. of particle 2, we obtain the transfer of T.K.E. from particle 1 to particle 2:

$$\text{T.K.E.}(1 \rightarrow 2) = (m/2)[n_2\vec{c}_{12} + (n_1 - n_2)(\vec{k}\cdot\vec{c}_{12})\vec{k} - n_2 d(\vec{k}\times\vec{\Omega})]\cdot(\vec{c}_1 + \vec{c}_2). \quad (1.14)$$

Similarly, the transfer of spin kinetic energy (S.K.E.) from particle 1 to particle 2 is found to be

$$\text{S.K.E.}(1 \rightarrow 2) = (mdn_2/4)[(\vec{k}\times\vec{c}_{12}) - d(\vec{k}\cdot\vec{\Omega})\vec{k} + d\vec{\Omega}]\cdot(\vec{\omega}_1 - \vec{\omega}_2). \quad (1.15)$$

Adding the changes in kinetic energies of particles 1 and 2 gives the total changes in translational and spin kinetic energies:

$$\begin{aligned}\Delta\text{T.K.E.} = m[&(n_1^2 - n_1)(\vec{k}\cdot\vec{c}_{12})^2 + (n_2^2 - n_2)(\vec{k}\times\vec{c}_{12})^2 \\ &- (2n_2^2 - n_2)d\vec{c}_{12}\cdot(\vec{k}\times\vec{\Omega}) + n_2^2 d^2(\vec{k}\times\vec{\Omega})^2], \quad (1.16)\\ \Delta\text{S.K.E.} = (mn_2^2/K)[&(\vec{k}\times\vec{c}_{12})^2 + 2d\vec{\Omega}\cdot(\vec{k}\times\vec{c}_{12}) + d^2(\vec{k}\times\vec{\Omega})^2] \\ &- mn_2[d\vec{\Omega}\cdot(\vec{k}\times\vec{c}_{12}) + d^2(\vec{k}\times\vec{\Omega})^2]. \quad (1.17)\end{aligned}$$

*Equation of state*

In the cell model, the particle is assumed to be enclosed in a spherical shell formed by the adjacent particles, and is moving with a velocity of order $v$ relative to the shell. If we define $s$ to be the average separation between particle surfaces, then the average collision rate is of order $v/s$. Referring to equation 1.12, if we set the relative velocity $c_{12}$ equal to $v$, and assume that the effects of spin on the pressure average to zero over all possible collisions (*i.e.*, $\overline{\vec{k}\times\vec{\Omega}} = 0$, there is on average no

correlation between the spin of the particle and the direction of the collision), then the momentum transferred in a typical collision is of order $mv$. Since the area of the cell is proportional to $d^2$, we have

$$p \sim mv\frac{1}{d^2}\frac{v}{s}.$$

If we collect all constant factors into a single dimensionless constant $t$, then

$$p = td\rho\frac{v^2}{s}. \tag{1.18}$$

*Coefficient of viscosity (translational)*

Here we consider the collision between particles from two adjacent shearing layers, moving with a relative velocity of $\Delta u$. In a shearing collision of this type, it is clear that particle spin will be a significant effect. However, in the linear momentum equation of motion the effect of spin has been separated out into the term involving $\eta_r$, and thus the constitutive relation for $\eta_t$ should not depend upon the surface roughness of the particles. Referring again to equation 1.12, and setting $\vec{c}_{12} = \Delta u$ and $n_2 = 0$, we see that the momentum transferred in a shearing collision is of order $m\Delta u$. Taking into account the collision rate and the area of the cell, the shear stress is of order:

$$\sigma \sim m\Delta u\frac{1}{d^2}\frac{v}{s}.$$

Since the grain layers have separations of order $d$, we can replace $\Delta u/d$ by $du/dy$. Combining all constants into $q_t$, we have

$$\sigma = q_t d^2 \rho\frac{v}{s}\frac{du}{dy}.$$

Comparing this to the term for translational shear stress in the linear momentum equations of motion, we find the expression for the translational coefficient of viscosity:

$$\eta_t = q_t d^2 \rho\frac{v}{s}. \tag{1.19}$$

*Coefficient of viscosity (rotational)*

To evaluate this coefficient, consider the case of a granular flow in which the

vorticity is non-zero and is not balanced by the average spin angular velocity ($2\vec{\omega} - \vec{\nabla} \times \vec{u} \neq 0$). By the momentum equation for spin (equation 1.7), this will result in a change in the average spin at a rate involving $\eta_r$. From equation 1.13, the change in spin of a particle in a typical collision is of order:

$$\Delta\vec{\omega} \sim \frac{md^2 n_2}{2I}(2\vec{\omega} - \vec{\nabla} \times \vec{u}).$$

Factoring in the volume of a cell and the rate of collision, and gathering all constant factors into $q_r$, the rate of change of spin angular momentum is

$$\rho\theta\frac{\Delta\vec{\omega}}{\Delta t} = -q_r\frac{m}{d^3}\frac{I}{m}\frac{md^2 n_2}{2I}(2\vec{\omega} - \vec{\nabla} \times \vec{u})\frac{v}{s}.$$

Comparing this expression with equation 1.7, the rotational coefficient of viscosity is found to be

$$\eta_r = q_r d^2 \rho \frac{v}{s} n_2. \tag{1.20}$$

*Coefficient of thermal diffusivity (translational)*

From equation 1.14 it is seen that the translational kinetic energy transfer in a typical collision is of order $mv\Delta v$. Multiplying this by the collision rate and dividing by the area, we obtain an expression for the translational kinetic energy flux:

$$Q_{(t)} \sim \frac{mv\Delta v}{d^2}\frac{v}{s},$$

or

$$Q_{(t)} = -r_t d^2 \frac{v}{s}\frac{d}{dy}(\frac{1}{2}\rho v^2).$$

Comparing this with the thermal diffusion term in equation 1.9 gives

$$K_t = r_t d^2 \frac{v}{s}. \tag{1.21}$$

*Coefficient of thermal diffusivity (rotational)*

From equation 1.15, the transfer of spin kinetic energy in a typical collision is of order $m(d^2/4)n_2 W\Delta W$. Thus, the flux of spin kinetic energy is

$$Q_{(r)} \sim \frac{md^2 n_2}{4} W\Delta W \frac{1}{d^2}\frac{v}{s},$$

or

$$Q_{(r)} = -r_r d^2 \frac{v}{s} \frac{d^2 n_2}{4\theta} \frac{d}{dy}(\frac{1}{2}\rho\theta W^2).$$

Comparing this with the thermal diffusion term in equation 1.11 gives

$$K_r = r_r \frac{d^4 n_2}{4\theta} \frac{v}{s}. \tag{1.22}$$

*Coefficient of translational thermal energy loss and transfer*

The term $I_t$ in the translational thermal energy equation includes the loss of energy due to inelasticity and surface friction effects and the transfer of energy to or from spin thermal energy. Its dependence upon the translational thermal velocity $v$ and the spin thermal velocity $W$ may be found by considering the expression for the total change in T.K.E. given by equation 1.16. If we assume that the relative velocity $c_{12}$ is of order $v$ and that the total spin $\Omega$ is of order $W$, then $I_t$ will consist of three terms depending upon $v^2$ and $W^2$. The first term on the right hand side of equation 1.16 represents the loss of energy due to inelasticity of the collision and is proportional to $v^2$. The second term represents the transfer of energy from translational thermal velocity to spin thermal velocity due to surface friction, and is also proportional to $v^2$. Under most circumstances this transfer will not be perfectly efficient, and this energy loss will be greater than the corresponding energy gain in the expression for $I_r$. The third term will average to zero over all collisions due to the assumption that the orientation of $\vec{\Omega}$ is uncorrelated to its amplitude or to the orientation of $\vec{k}$ or $\vec{c}_{12}$. Finally, the fourth term represents the transfer of energy from spin thermal velocity to translational thermal velocity due to surface friction and is proportional to $W^2$. Once again, inefficiencies in this process mean that this term will be less than the corresponding term in $I_r$.

Factoring in the rate of collisions and the volume of the cell, and gathering the constant factors into one constant for each term, we have

$$I_t = \gamma_t \rho \frac{v^3}{s} + \gamma_t' \rho \frac{v^3}{s} - \gamma_t'' \rho \frac{d^2 W^2}{4} \frac{v}{s}. \tag{1.23}$$

*Coefficient of rotational thermal energy loss and transfer*

The determination of the form of $I_r$ closely follows that of $I_t$, using equation

1.17. The only significant difference is that spin thermal energy is not lost due to the inelasticity of collisions. By the same arguments as above, the form of $I_r$ is

$$I_r = -\gamma_r' \rho \frac{v^3}{s} + \gamma_r'' \rho \frac{d^2 W^2}{4} \frac{v}{s}. \tag{1.24}$$

## Solutions to the Equations of Motion

*Uniformly excited system*

In order to demonstrate the time evolution of the thermal velocities, we will first consider the problem of a spatially uniform system with arbitrary initial thermal velocities. To simplify the problem, the average flow velocity and average spin velocity are set to zero, as are all spatial derivatives. The energy equations now reduce to a pair of coupled, nonlinear, first order differential equations:

$$\frac{\partial(v^2)}{\partial t} = -2(\gamma_t + \gamma_t')\frac{v^3}{s} + 2\gamma_t'' \frac{d^2 W^2}{4} \frac{v}{s},$$

$$\frac{\partial}{\partial t}\left(\frac{d^2 W^2}{4}\right) = \frac{d^2}{4\theta} 2\gamma_r' \frac{v^3}{s} - \frac{d^2}{4\theta} 2\gamma_r'' \frac{d^2 W^2}{4} \frac{v}{s}.$$

It is expected that the translational and spin thermal velocities will exchange energy until some type of "equipartition" is reached. Also, both thermal velocities should decay with time due to inelasticity and surface friction losses.

An analytical solution to these equations is possible in the special case that the ratio of spin thermal velocity to translational thermal velocity is a constant determined by the particle properties:

$$v(t) = \frac{v_0}{(bv_0 t + 1)} \quad \text{and} \quad W(t) = \frac{W_0}{(bv_0 t + 1)},$$

$$\frac{d^2 W^2}{4v^2} = \left\{ \gamma_t + \gamma_t' - \frac{d^2}{4\theta}\gamma_r'' + \left[ \left( \gamma_t + \gamma_t' - \frac{d^2}{4\theta}\gamma_r'' \right)^2 + \frac{d^2}{\theta}\gamma_r'\gamma_t'' \right]^{\frac{1}{2}} \right\} / 2\gamma_t'' \equiv a^2, \tag{1.25}$$

$$b = \frac{1}{s}[\gamma_t + \gamma_t' - \gamma_t'' a^2],$$

where $v_0$ is the initial translational thermal velocity, and $W_0 = 2av_0/d$. However, a general analytical solution for arbitrary initial conditions is not known.

The solutions for different initial conditions were obtained by numerical methods and are shown in figures 1.3a, 1.3b, and 1.3c. In the first case, the initial translational and spin thermal velocities were set to the ratio "$a$" given above. It can be seen that this ratio is maintained as the thermal velocities decay. For the second case, the initial spin thermal velocity was set to zero. For the third case, the initial translational thermal velocity was set to a small value. (If it were set to zero, nothing would happen since no particle collisions would occur.)

In all cases, it can be seen that the translational and spin thermal velocities tend to exchange energy until their ratio reaches the value given in equation 1.25.

*Steady state system with no flow*

This problem is intended to illustrate the spatial variation in translational and spin thermal velocities as energy is supplied from a wall under steady state conditions. The grains are confined between two infinite parallel plates, with no average spin or flow velocities and no gravitational forces (figure 1.4). The plates are vibrating so as to supply thermal energy to the grains; however, the motion of the plates is assumed not to produce spatially correlated spin or linear motion in the grains. Inasmuch as a theory for the boundary conditions for granular flows has not been developed to include spin, the values of the translational and spin thermal velocities will be arbitrarily set at the walls.

From the linear momentum equation, the pressure is found to be uniform throughout the region between the plates. Setting all time derivatives to zero, the energy equations reduce to a pair of coupled, nonlinear, second order differential equations:

$$0 = \frac{\partial}{\partial x}\left[r_t d^2 \frac{v}{s}\frac{\partial}{\partial x}(\frac{1}{2}\rho v^2)\right] - \gamma_t \rho \frac{v^3}{s} - \gamma_t' \rho \frac{v^3}{s} + \gamma_t'' \rho \frac{d^2 W^2}{4}\frac{v}{s},$$

$$0 = \frac{\partial}{\partial x}\left[r_r \frac{d^4}{4}\frac{n_2}{\theta}\frac{v}{s}\frac{\partial}{\partial x}(\frac{1}{2}\rho\theta W^2)\right] + \gamma_r' \rho \frac{v^3}{s} - \gamma_r'' \rho \frac{d^2 W^2}{4}\frac{v}{s}.$$

Substituting $p_0 = td\rho v^2/s$ for the uniform pressure gives

$$0 = r_t d^2 \frac{\partial}{\partial x}\left(p_0 \frac{\partial v}{\partial x}\right) - (\gamma_t + \gamma_t')p_0 v + \gamma_t'' p_0 \frac{d^2 W}{4v} W,$$

$$0 = r_r \frac{d^4 n_2}{4} \frac{\partial}{\partial x}\left(p_0 \frac{W}{v}\frac{\partial W}{\partial x}\right) + \gamma_r' p_0 v - \gamma_r'' p_0 \frac{d^2 W}{4v} W.$$

Once again, an analytical solution to these equations is possible if the ratio of spin thermal velocity to translational thermal velocity is assumed to be a definite constant:

$$v(x) = v_0[\exp(x/\lambda) + \exp(-x/\lambda)]/2,$$

$$W(x) = W_0[\exp(x/\lambda) + \exp(-x/\lambda)]/2,$$

$$\frac{d^2 W^2}{4v^2} = \frac{r_r n_2(\gamma_t + \gamma_t') - r_t \gamma_r'' + \left[(r_r n_2(\gamma_t + \gamma_t') - r_t \gamma_r'')^2 + 4 r_t \gamma_r' \gamma_t'' r_r n_2\right]^{\frac{1}{2}}}{2\gamma_t'' r_r n_2} \equiv a^2, \tag{1.26}$$

$$\lambda^2 = \frac{r_t d^2}{(\gamma_t + \gamma_t' - \gamma_t'' a^2)},$$

where $v_0$ = translational thermal velocity at the center, and $W_0 = 2av_0/d$. An analytical solution for arbitrary boundary conditions is not known.

Numerical solutions for various boundary conditions are shown in figures 1.5a, 1.5b, and 1.5c. In figure 1.5a, the thermal velocity amplitudes are plotted for the case in which the ratio of spin to translational thermal velocity at the wall was set to the value "$a$" given in (1.26). In figure 1.5b, the spin thermal velocity at the wall was set to zero. In figure 1.5c, the translational thermal velocity at the wall was set to a small value. The ratio of spin to translational thermal velocity is seen to tend towards a fixed ratio as distance from the wall increases. This ratio, as given in equation 1.26, is a function of particle and collision parameters only, and is independent of the thermal velocities. Physically, the initial increase in spin (or translational) thermal velocity near the walls in figure 1.5b (or figure 1.5c) is due to the large difference in energy between these two modes created by the wall boundary conditions. As we move away from the wall, energy is exchanged during collisions until the equilibrium ratio of thermal velocities is achieved. The ensuing uniform decay of these velocities towards the center is due to the combined effects of diffusion and loss of thermal energy.

*Steady state Couette flow*

Here we consider an extension of the previous problem in which translational and spin thermal energy can be generated within the flow by shearing motion. The

grains are again confined between two parallel plates with no gravitational forces acting. Figure 6 illustrates the coordinate system and the motion of the two plates; the system is of infinite extent in the $x$ and $z$ directions. In order to take advantage of the symmetry of the system, the plates are moving with equal but opposite velocities; thus the average flow velocity is set to zero at $y = 0$. The average flow velocity is assumed to be parallel to the $x$-direction and all flow variables are taken to depend only on $y$.

Consideration of the spin angular momentum equation (equation 1.7) reveals that the $x$- and $y$-components of the average spin angular velocity will be zero, while the $z$-component will be given by

$$\omega_z = -\frac{1}{2}\frac{\partial u_x}{\partial y}. \tag{1.27}$$

The $y$-component of the linear momentum equation gives $dp/dy = 0$ so that the pressure in the channel is constant:

$$p = p_0 = td\rho\frac{v^2}{s}. \tag{1.28}$$

The $x$-component of the linear momentum equation shows that the shear stress is also a constant:

$$\sigma = \sigma_0 = \eta_t\frac{\partial u_x}{\partial y}. \tag{1.29}$$

Substitution of equation 1.27 into the thermal energy equations gives

$$0 = -\frac{\partial}{\partial y}\left[-K_t\frac{\partial}{\partial y}(\frac{1}{2}\rho v^2)\right] - I_t + \eta_t\frac{\partial u_x}{\partial y},$$

$$0 = -\frac{\partial}{\partial y}\left[-K_r\frac{\partial}{\partial y}(\frac{1}{2}\rho\theta W^2)\right] - I_r.$$

Using the expressions for $\eta_t$, $K_t$, $K_r$, $I_t$, and $I_r$ (equations 1.19, and 1.21 to 1.24), and making use of the constant pressure (equation 1.28) and shear stress (equation 1.29), we obtain two coupled, non-linear, second order differential equations for $v$ and $W$:

$$0 = \frac{\partial^2 v}{\partial y^2} + \left[-\frac{(\gamma_t + \gamma_t')}{r_t d^2} + \frac{\gamma_t''}{r_t d^2}\left(\frac{d^2W^2}{4v^2}\right) + \frac{t^2\sigma_0^2}{r_t q_t d^2 p_0^2}\right] v, \tag{1.30}$$

$$0 = \frac{\partial}{\partial y}\left[\frac{dW}{2v}\frac{\partial}{\partial y}\left(\frac{dW}{2}\right)\right] + \left[\frac{\gamma_r'}{r_r d^2} - \frac{\gamma_r''}{r_r d^2}\left(\frac{d^2W^2}{4v^2}\right)\right]\frac{v}{n_2}. \tag{1.31}$$

Since a theory of the boundary conditions at a wall including spin is not available, we will arbitrarily set the values of the translational and spin thermal velocities at the wall and consider two different examples, which will be solved by numerical methods.

In the first case, the wall properties have been chosen so that the spin thermal velocity at the wall is zero. The wall velocity is adjusted so as to obtain a desired value of the translational thermal velocity at the wall. The particle properties have been chosen so as to obtain a shear stress to pressure ratio of

$$\frac{\sigma_0}{p_0} > \frac{\sqrt{q_t}}{t}\left(\gamma_t + \gamma_t' - \frac{\gamma_t''\gamma_r'}{\gamma_r''}\right).$$

With this ratio of shear stress to pressure, the rate of energy generation in the channel due to shearing exceeds the rate of energy loss due to friction and inelasticity; the excess energy diffuses to the walls where it is absorbed. As in the examples given in the last section, the spin thermal velocity gains energy from the translational thermal velocity as we move away from the wall, until a definite ratio is achieved (figure 1.7a). This ratio can be found by analytically solving equations 1.30 and 1.31 with the condition that $W/v$ is a constant:

$$\frac{d^2W^2}{4v^2} =$$

$$\left\{(\gamma_t + \gamma_t') - \frac{t^2\sigma_0^2}{q_t p_0^2} - \frac{r_t\gamma_r''}{r_r n_2} + \left[\left((\gamma_t + \gamma_t') - \frac{t^2\sigma_0^2}{q_t p_0^2} - \frac{r_t\gamma_r''}{r_r n_2}\right)^2 + \frac{4r_t\gamma_r'\gamma_t''}{r_r n_2}\right]^{\frac{1}{2}}\right\}/2\gamma_t''. \tag{1.32}$$

The average flow velocity is shown in figure 1.7b and the average spin velocity is shown in figure 1.7c.

In the second case, the translational thermal velocity has been set to a very small value at the wall. As shown in figure 1.8a, the dip in spin thermal velocity as distance from the wall increases is due to transfer of energy to translational thermal velocity, until the ratio given in equation 1.32 is reached. The average flow velocity and average spin velocity are given in figures 1.8b and 1.8c.

It is interesting to note that the average spin velocity is in fact proportional to the translational thermal velocity. This can be shown by combining equations 1.19, 1.27, 1.28, and 1.29 to obtain

$$\omega_z = \frac{t}{2q_t}\frac{\sigma_0}{p_0}\frac{v}{d}.$$

*Couette flow with and without particle spin*

It would be of interest to compare the theory presented here with a similar theory of granular flow which does not include particle spin (Haff (1983)). For this purpose, the problem of steady state Couette flow with no gravity will be considered, with attention given to a comparison of pressure, shear stress, and flow velocity with and without the effects of particle spin.

In general, the pressure and shear stress will be determined to a significant degree by the boundary conditions on the translational and spin thermal velocities. Lacking a theory for these boundary conditions, the following two assumptions will be made: (1) the special case of a constant ratio between the translational and spin thermal velocities will be used and (2) the characteristic length over which the translational thermal velocity changes ($\lambda$) will be held constant. If the rate of shearing is high enough such that the rate of thermal energy generation is greater than the rate of dissipation, then the thermal velocities will be given by

$$v(y) = v_0 \cos(y/\lambda), \qquad W(y) = W_0 \cos(y/\lambda),$$

and the ratio of spin to translational thermal velocity can be found from equation 1.31:

$$\left(\frac{dW}{2v}\right)^2 = \frac{\gamma_r'}{\gamma_r'' + r_r n_2 d^2/\lambda^2}. \tag{1.33}$$

To find the pressure $p_0$, information concerning the total amount of free volume in the flow is needed. For this purpose, the free space parameter $\Delta h$ is introduced, and is defined as the difference between the total thickness of the grains (when close packed) and the width of the channel (figure 1.9). The pressure can then be determined by integrating the average particle to particle spacing "$s$" across the channel and relating this to $\Delta h$ (Haff (1983)):

$$p_0 = 3t\rho v_0^2 \left[\frac{\lambda}{\Delta h}\sin\left(\frac{h}{2\lambda}\right)\cos\left(\frac{h}{2\lambda}\right) + \frac{h}{2\Delta h}\right].$$

From this expression, it can be seen that if the value of $\lambda$ is held constant, the pressure in the flow does not depend upon the effects of particle spin.

The ratio of shear stress to pressure can be found from equation 1.30:

$$\frac{\sigma_0}{p_0} = \frac{\sqrt{q_t}}{t}\left[\gamma_t - \frac{r_t d^2}{\lambda^2} + \frac{1}{\gamma_r'' + r_r n_2 d^2/\lambda^2}\left(\gamma_t' r_r n_2 \frac{d^2}{\lambda^2} + \gamma_t'\gamma_r'' - \gamma_t''\gamma_r'\right)\right]^{\frac{1}{2}}. \quad (1.34)$$

Since energy will normally be lost in the transfers between translational and spin thermal energy, the following inequalities will hold:

$$\gamma_t' > \gamma_r', \quad \gamma_r'' > \gamma_t'', \quad \gamma_t'\gamma_r'' - \gamma_t''\gamma_r' > 0.$$

Using the last of these inequalities in equation 1.34, it can be seen that the effect of adding spin and surface roughness to the theory is to increase the ratio of shear stress to pressure. Since the pressure is unchanged, the shear stress will increase.

Finally, the flow velocity at the wall may be found by integrating equation 1.29:

$$u(h/2) = \frac{t}{q_t}\frac{\sigma_0}{p_0}\frac{\lambda}{d} v_0 \sin\left(\frac{h}{2\lambda}\right).$$

Holding the translational thermal velocity at the wall constant, the flow velocity will increase in proportion to the ratio of shear stress to pressure.

Thus it can be seen that in order to allow for the increased loss of energy due to surface friction, while keeping the thermal velocity profile constant, it is necessary to increase the rate of energy input to the flow by increasing both the shear stress and the shear velocity when spin is included in the theory.

**Figure Captions**

Figure 1.1: Energy flow relationships for the present model of granular flow including spin.

Figure 1.2: Diagram of the model of a binary collision, showing the particle diameter ($d$), particle spins ($\vec{\omega}_1$ and $\vec{\omega}_2$), the relative velocity ($\vec{c}_{12}$), and the unit vector ($\vec{k}$) between the particle centers at collision.

Figure 1.3: Evolution of translational and spin thermal velocities with time in a spatially uniform system: (a) $W(0) = 2av(0)/d$, (b) $W(0) = 0$, (c) $v(0)$ is small. (Velocities in arbitrary units.)

Figure 1.4: Illustration of grains confined between parallel vibrating plates.

Figure 1.5: Variation of translational and spin thermal velocities of grains confined between parallel vibrating plates: (a) $W(walls) = 2av(wall)/d$, (b) $W(walls) = 0$, (c) $v(wall)$ is very small. (Velocities in arbitrary units.)

Figure 1.6: Illustration of Couette flow geometry.

Figure 1.7: Variation in granular flow properties in Couette flow with the spin thermal velocity at the wall set to zero: (a) the thermal velocities, (b) the average flow velocity, (c) the average spin velocity. (Thermal velocities are in arbitrary units.)

Figure 1.8: Variation in granular flow properties in Couette flow with the translational thermal velocity at the wall set to a very small value: (a) the thermal velocities, (b) the average flow velocity, (c) the average spin velocity. (Thermal velocities are in arbitrary units.)

Figure 1.9: Illustration of the definition of the free space parameter $\Delta h$.

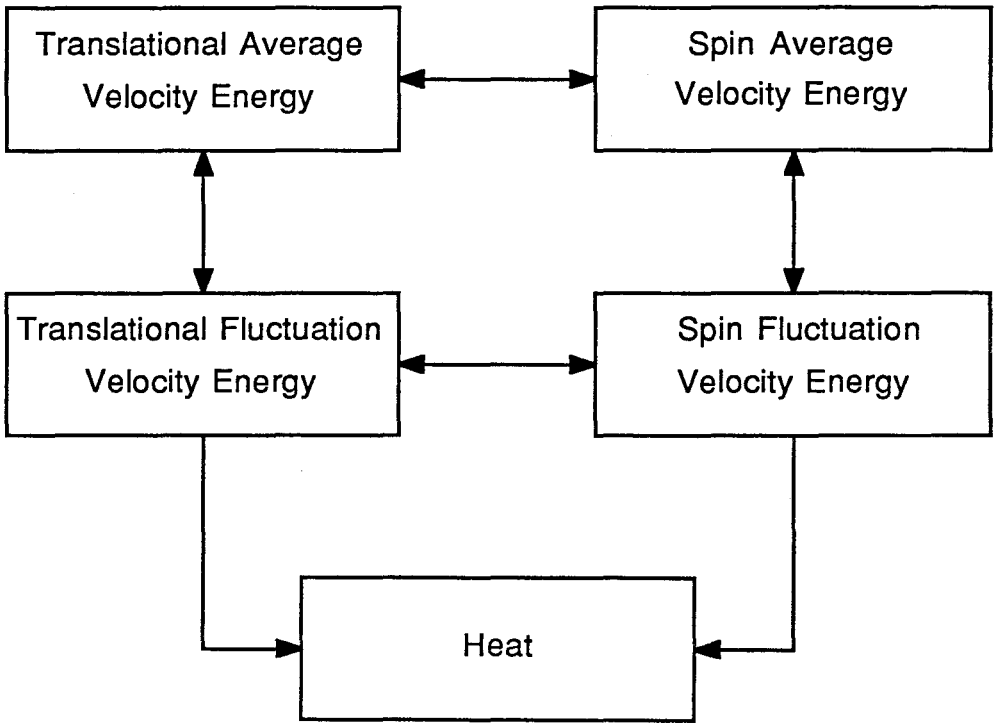

Translational Average
Velocity Energy
Spin Average
Velocity Energy
Translational Fluctuation
Velocity Energy
Spin Fluctuation
Velocity Energy
Heat

Figure 1.1

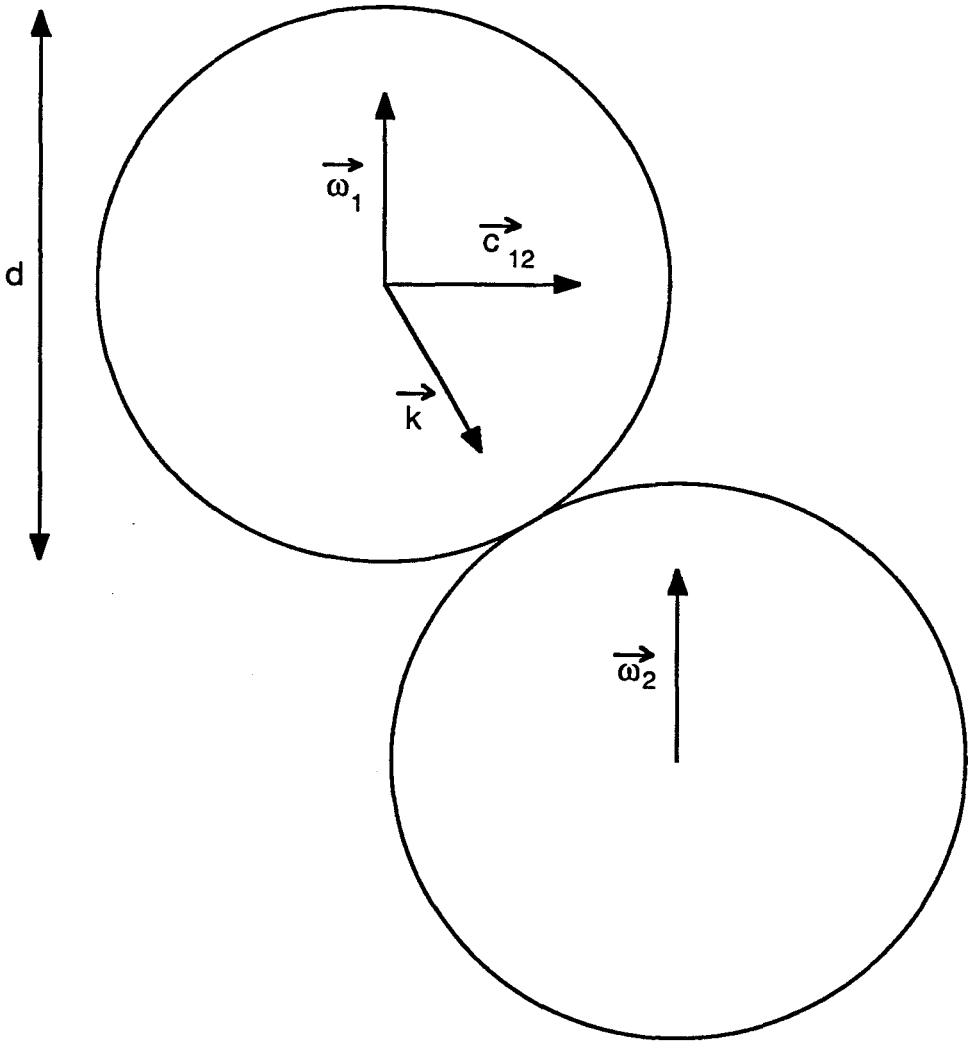



Figure 1.2

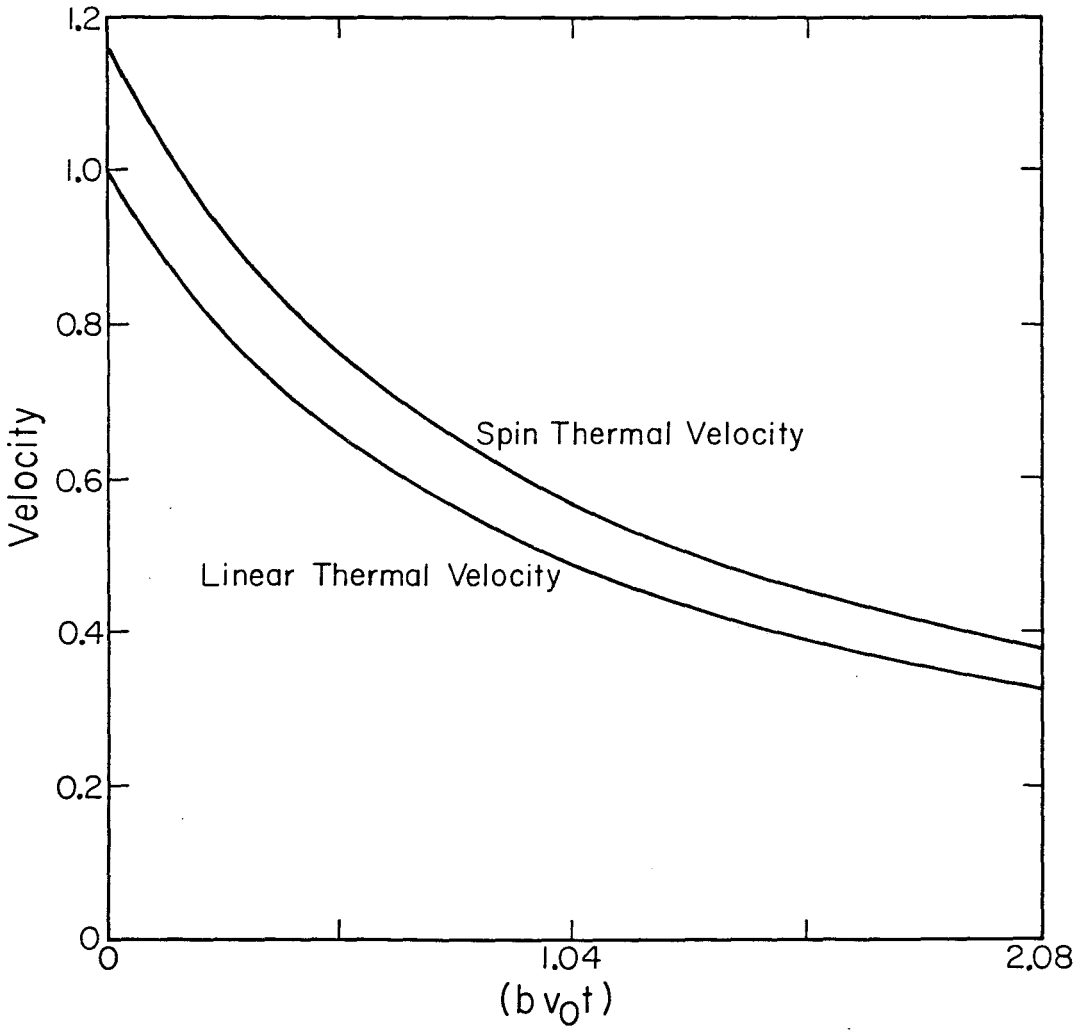

1.2
1.0
0.8
0.6
0.4
0.2
0
Velocity
Spin Thermal Velocity
Linear Thermal Velocity
0
1.04
2.08
(b v0 t)

Figure 1.3a

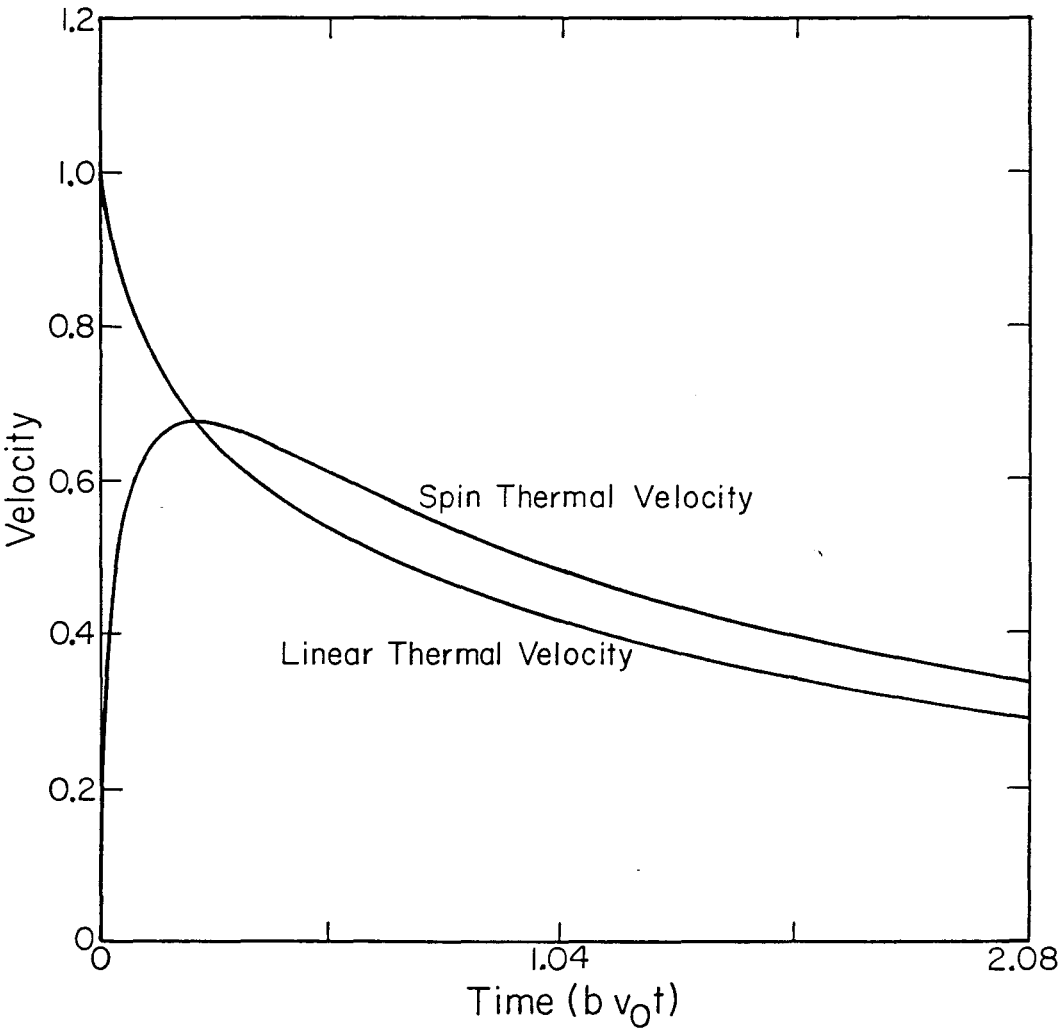

1.2
1.0
0.8
0.6
0.4
0.2
0
Velocity
Spin Thermal Velocity
Linear Thermal Velocity
0
1.04
2.08
Time (b $v_0$t)

Figure 1.3b

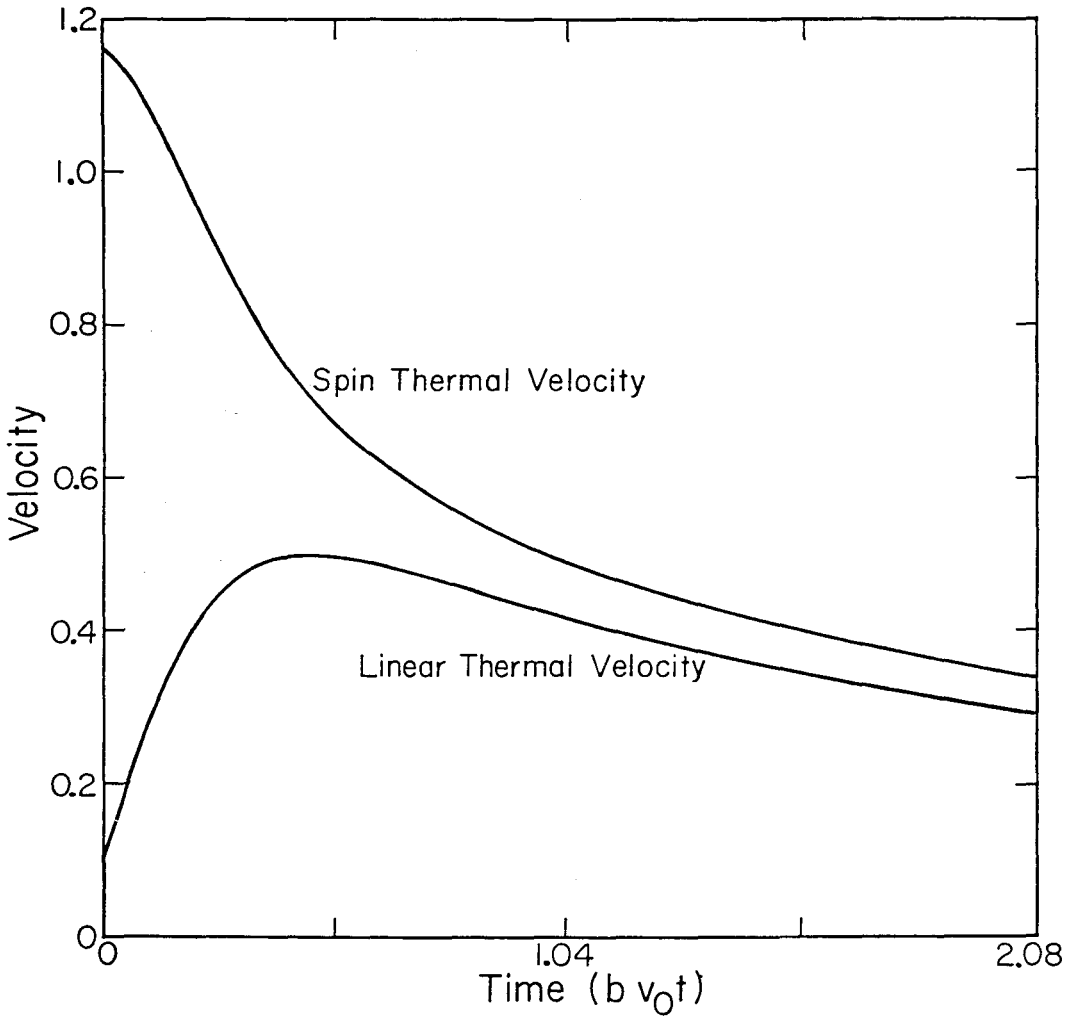

1.2
1.0
0.8
0.6
0.4
0.2
0
Velocity
Spin Thermal Velocity
Linear Thermal Velocity
0
1.04
2.08
Time (b $v_0$t)

Figure 1.3c

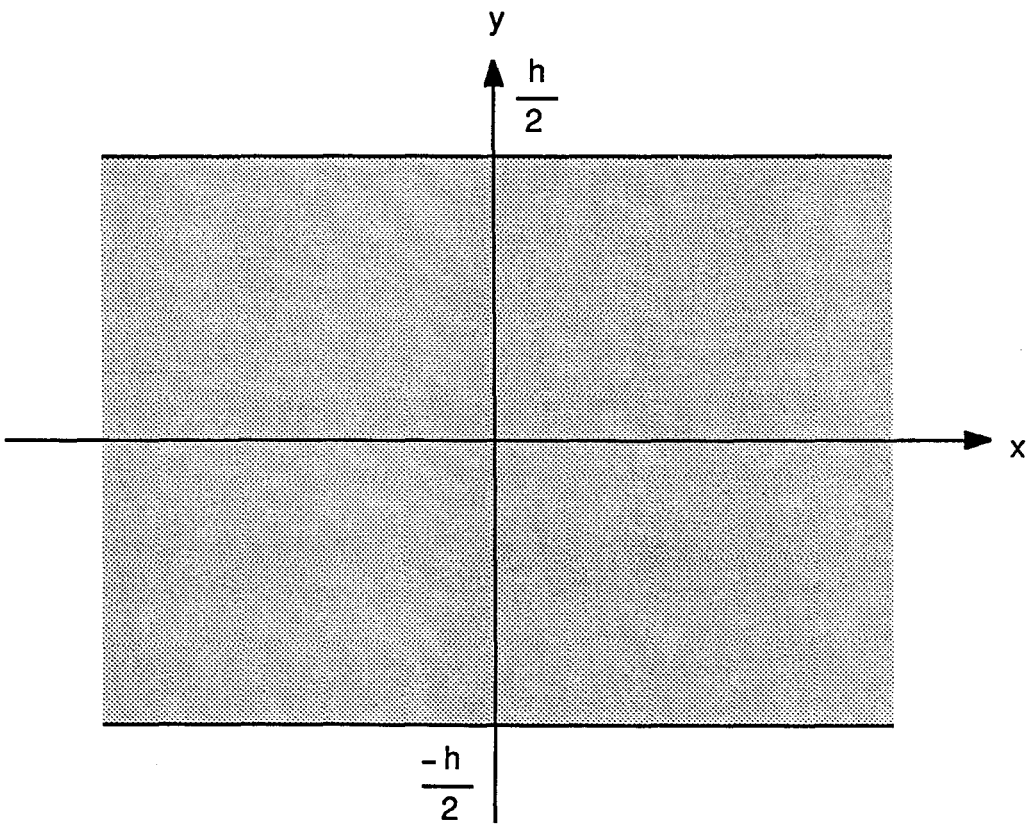



Figure 1.4

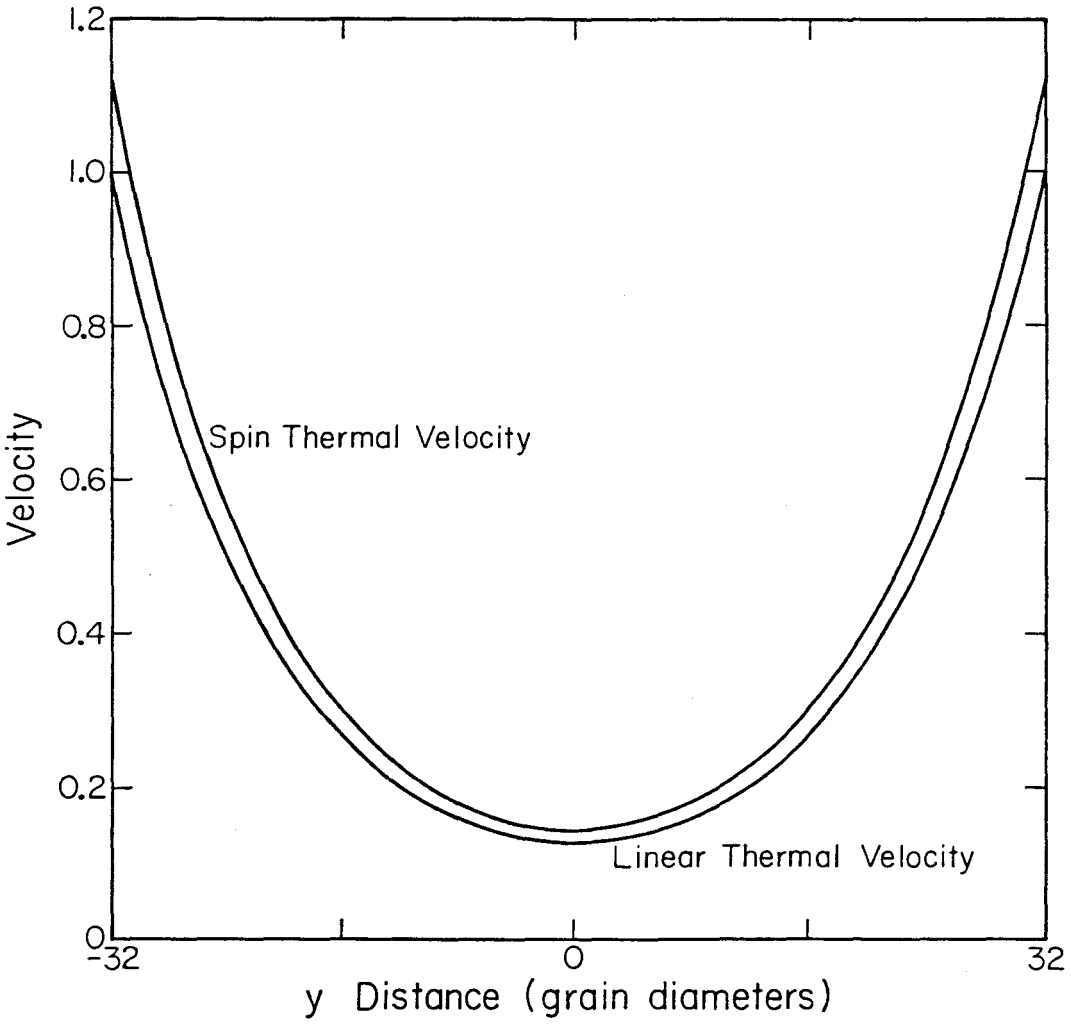

1.2
1.0
0.8
0.6
0.4
0.2
0
Velocity
Spin Thermal Velocity
Linear Thermal Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.5a

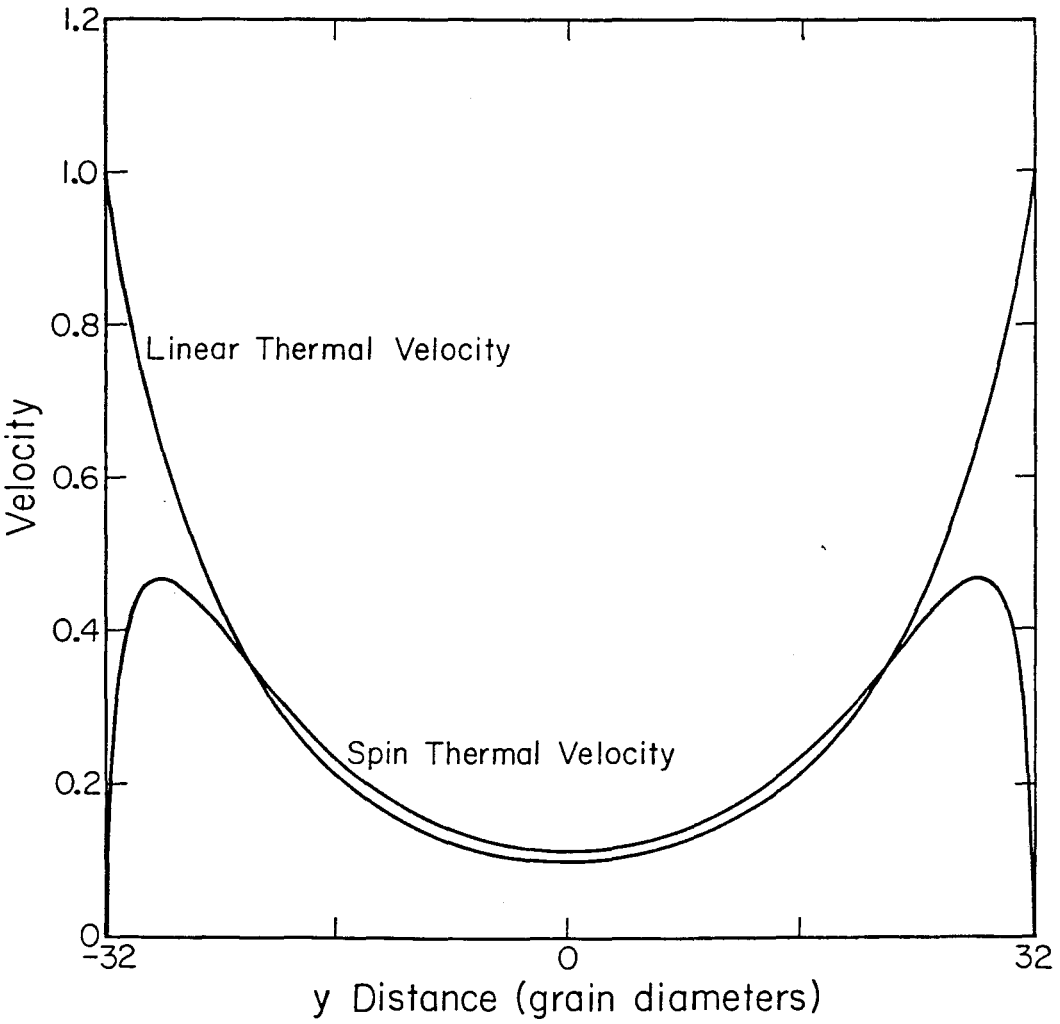

1.2
1.0
0.8
0.6
0.4
0.2
0
Velocity
Linear Thermal Velocity
Spin Thermal Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.5b

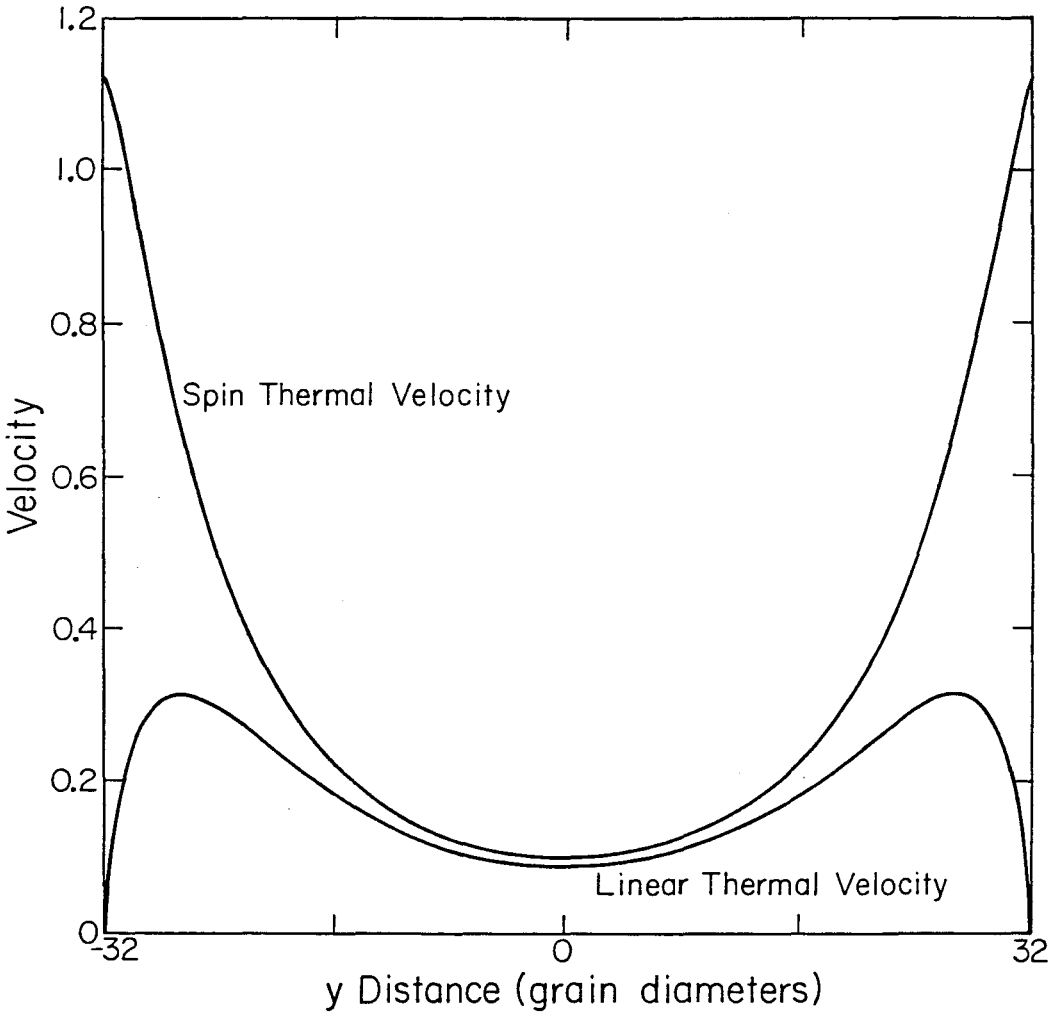

1.2
1.0
0.8
0.6
0.4
0.2
0
Velocity
Spin Thermal Velocity
Linear Thermal Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.5c

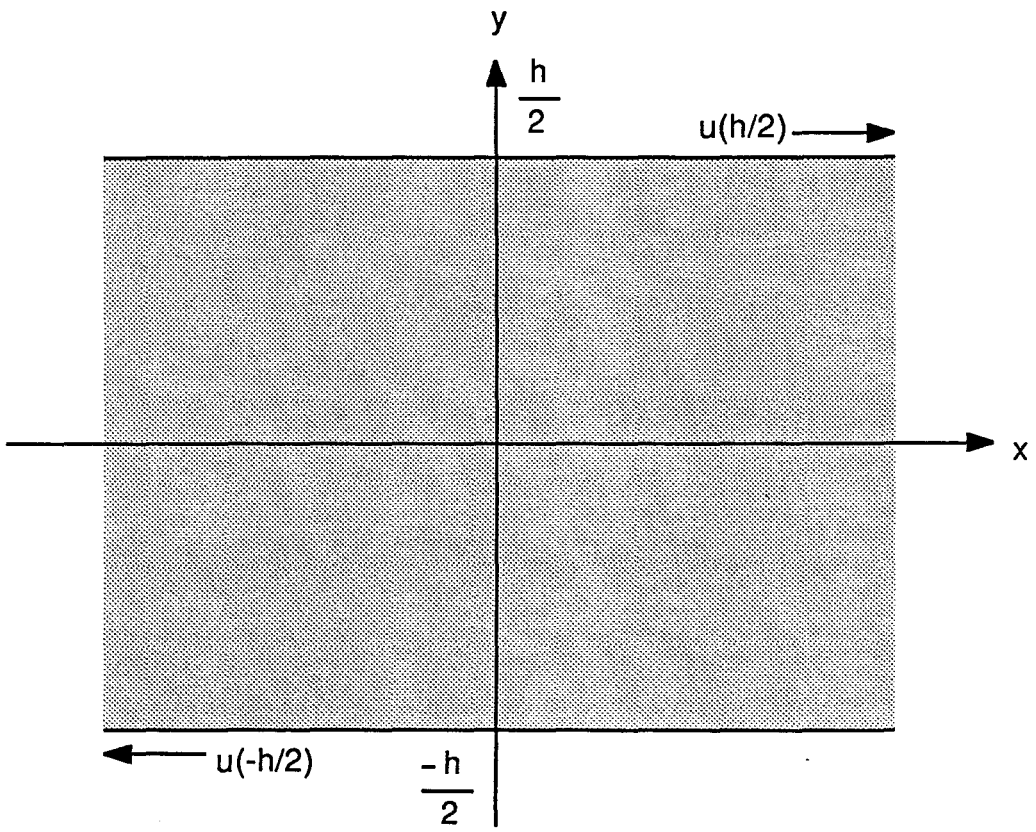



Figure 1.6

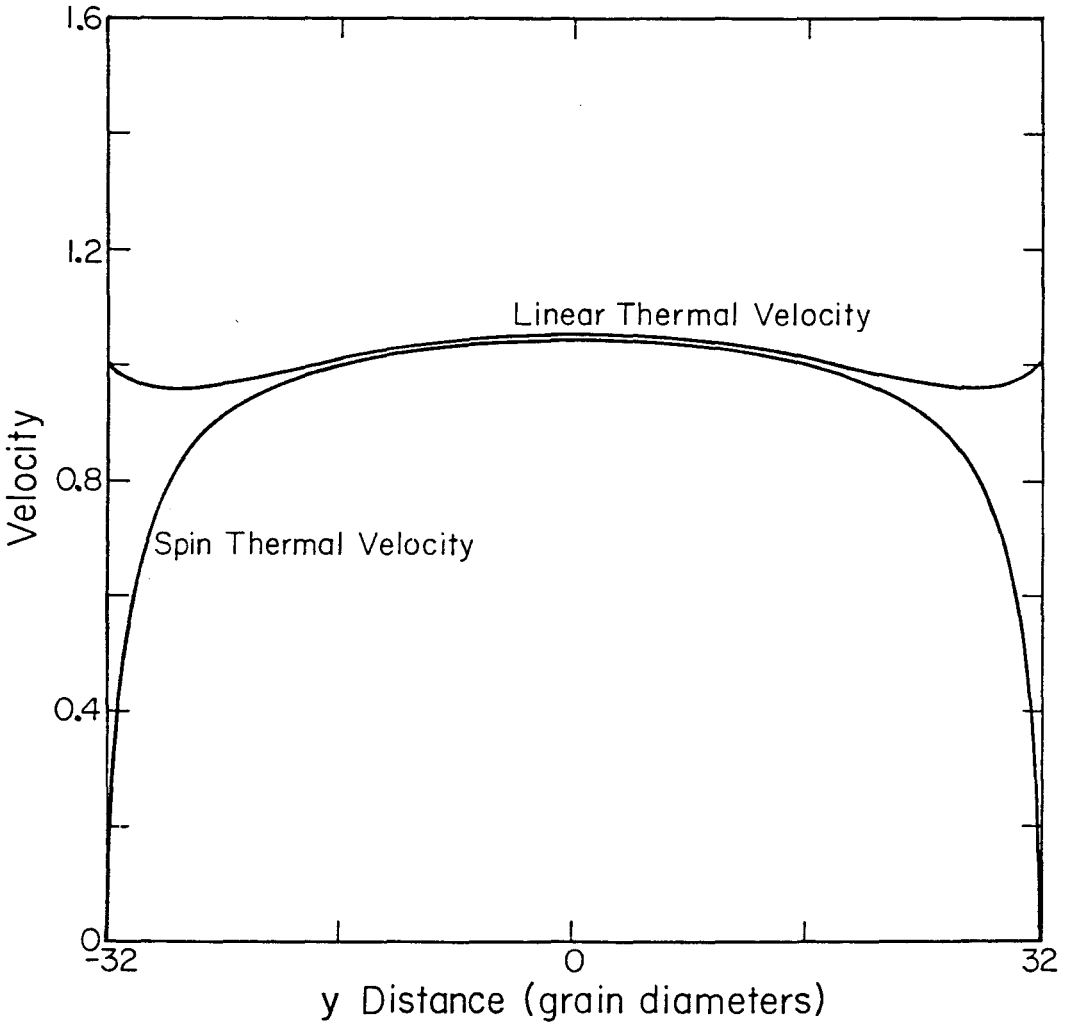

1.6
1.2
0.8
0.4
0
Velocity
Linear Thermal Velocity
Spin Thermal Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.7a

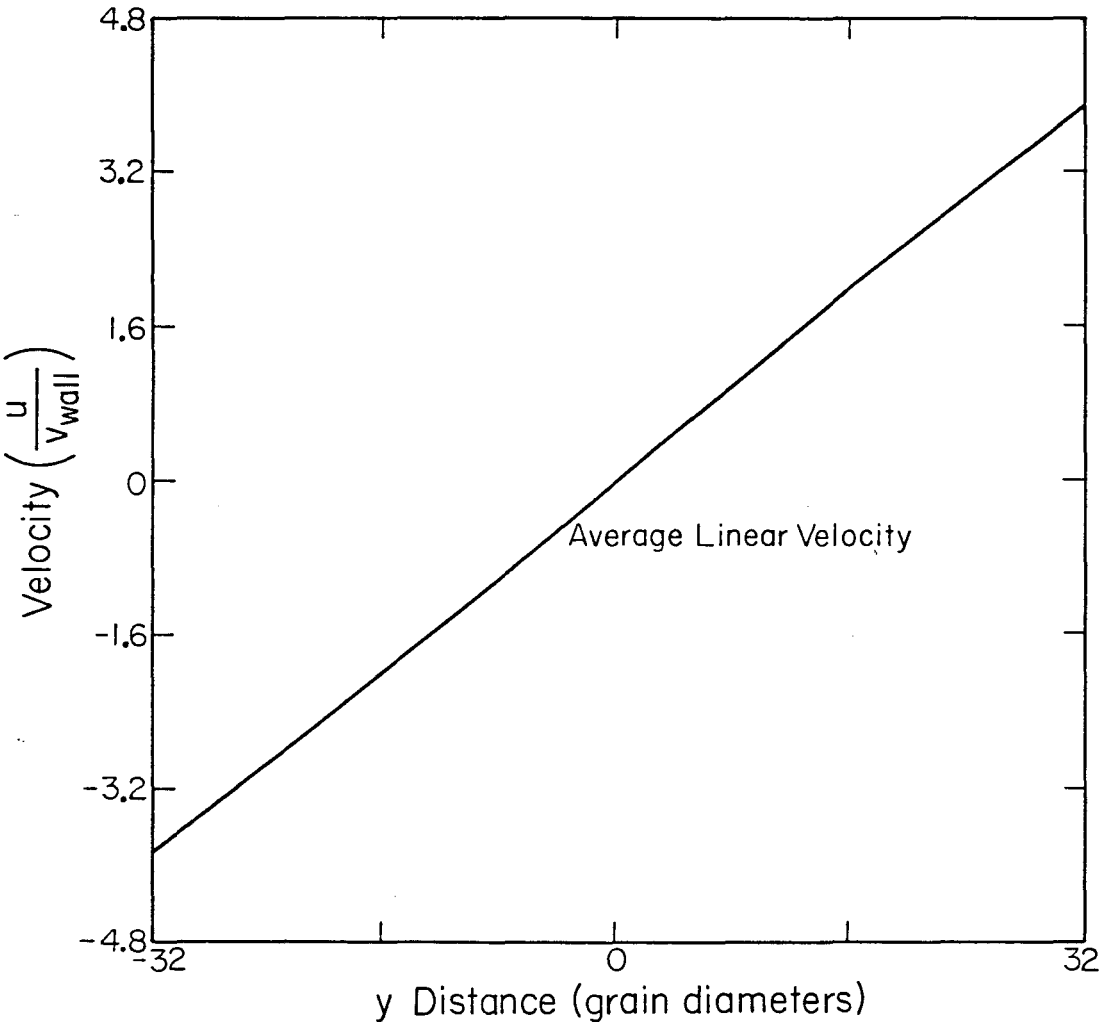

4.8
3.2
1.6
0
-1.6
-3.2
-4.8
-32
0
32
Velocity $\left(\frac{u}{V_{wall}}\right)$
Average Linear Velocity
y Distance (grain diameters)

Figure 1.7b

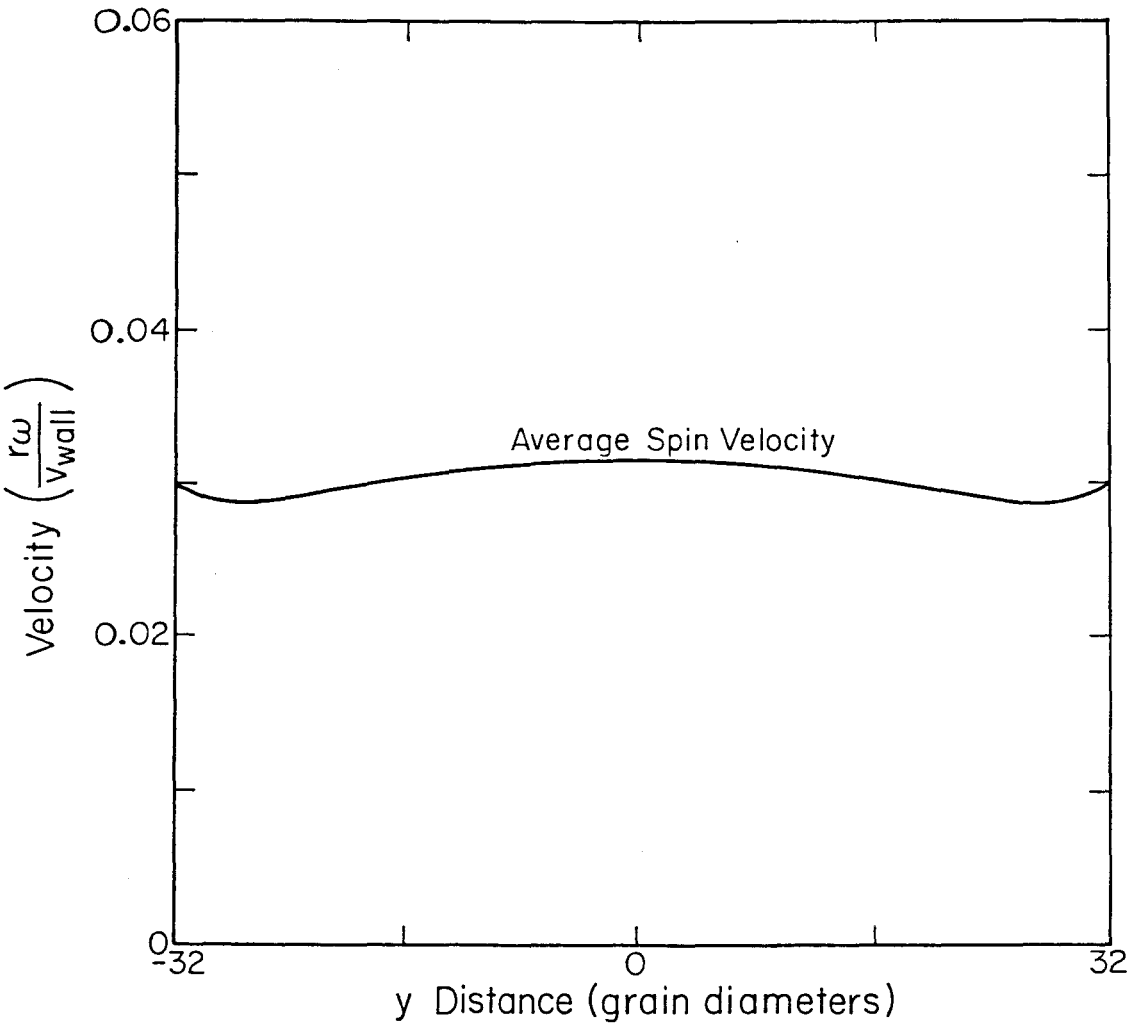

0.06
0.04
0.02
0
Velocity $\left(\frac{r\omega}{V_{wall}}\right)$
Average Spin Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.7c

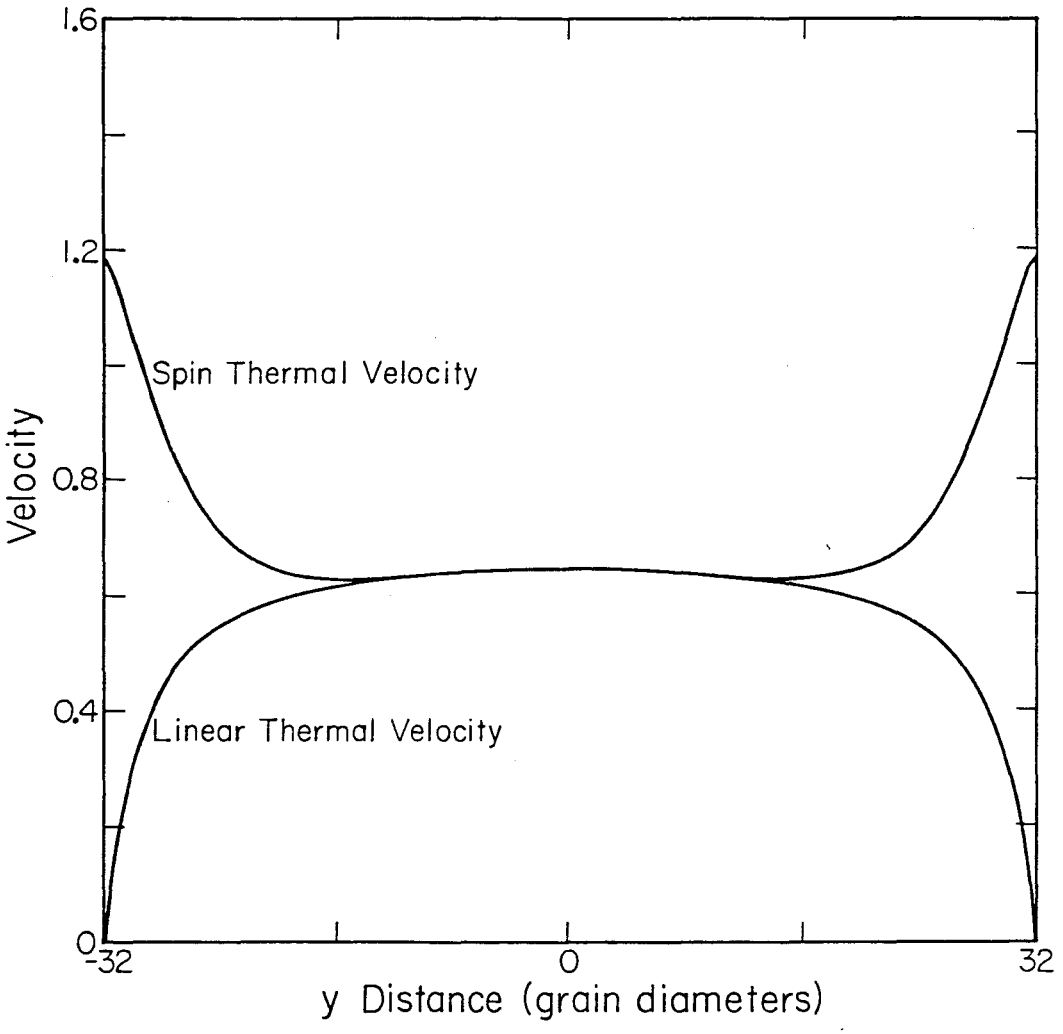

1.6
1.2
0.8
0.4
0
Velocity
Spin Thermal Velocity
Linear Thermal Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.8a

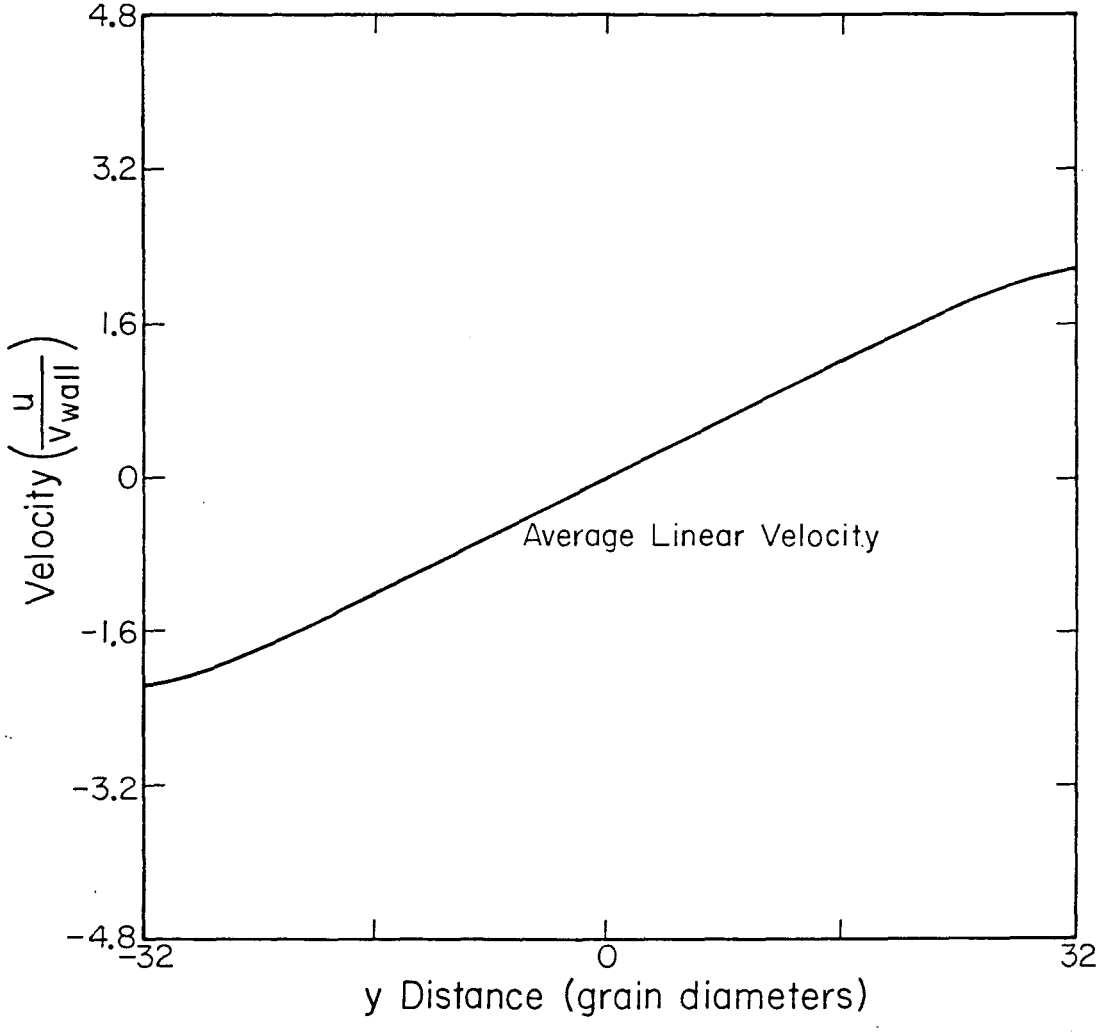

4.8
3.2
1.6
0
-1.6
-3.2
-4.8
Velocity $\left(\frac{u}{V_{wall}}\right)$
Average Linear Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.8b

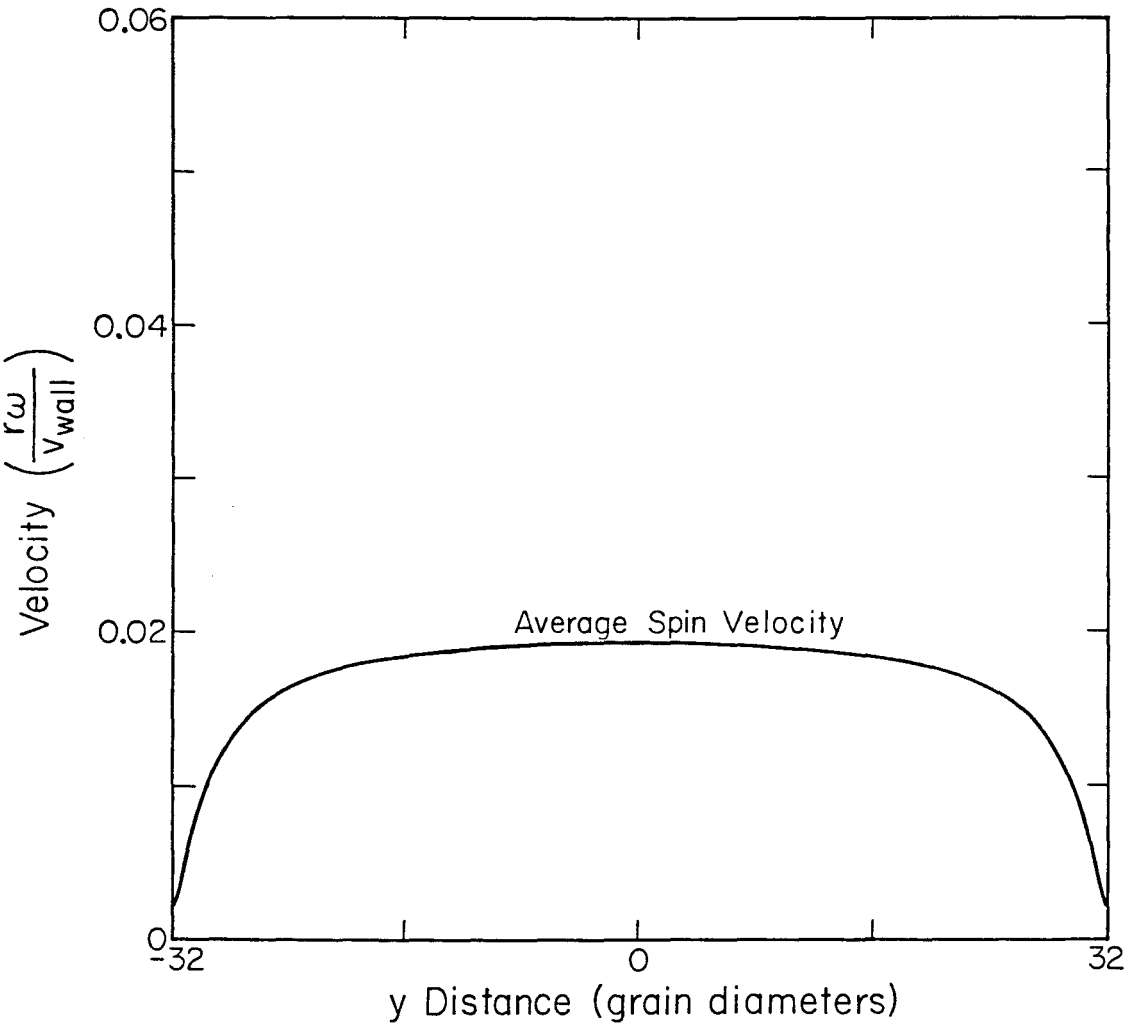

0.06
0.04
0.02
0
Velocity $\left(\frac{r\omega}{v_{wall}}\right)$
Average Spin Velocity
-32
0
32
y Distance (grain diameters)

Figure 1.8c

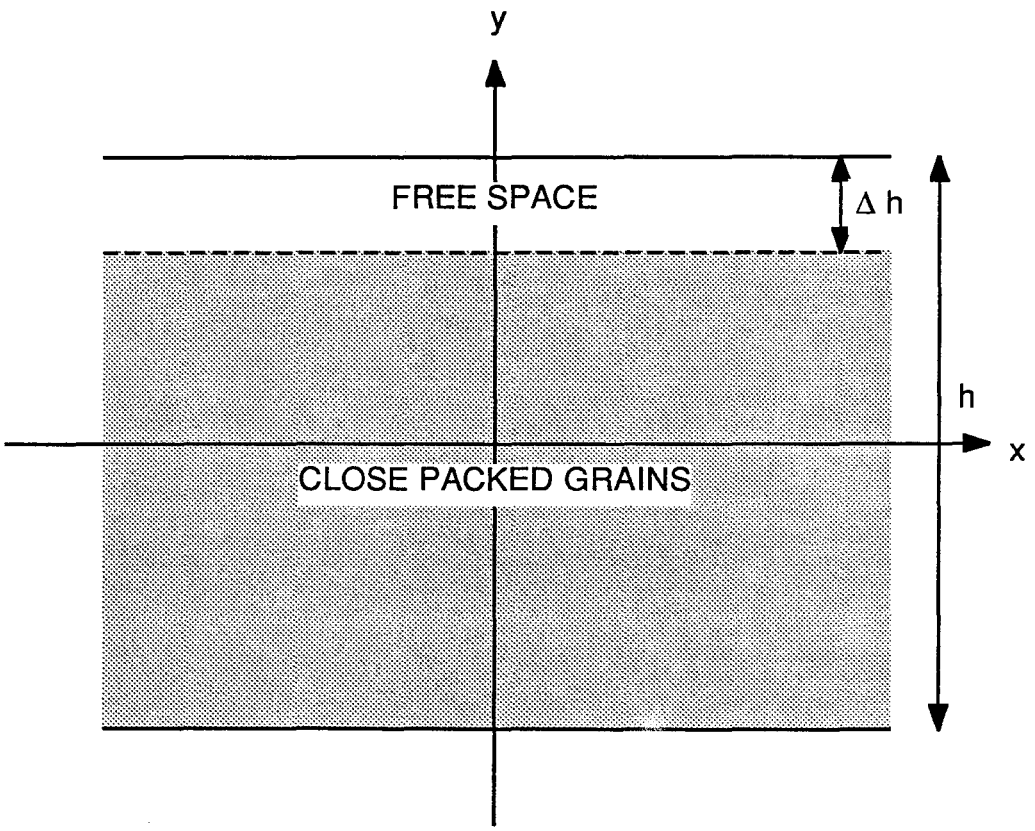



Figure 1.9

# Chapter 2: Boundary Conditions on Continuum Theories of Granular Flow

In order to solve practical problems using continuum theories of granular flow, boundary conditions are required which relate the parameters of the granular system adjacent to a wall to the forces and velocities associated with that wall. Therefore, in this chapter we will consider the development of such boundary conditions. As will be seen below, these boundary conditions will be significantly more complex than the boundary conditions of fluid mechanics; consequently, we will not include the additional complications associated with rough, spinning grains. Instead, we will work with the simpler continuum theories which have been presented for systems of smooth, non-spinning, inelastic spheres (Ogawa *et al.* (1980), Haff (1983), Jenkins and Savage (1983), Lun *et al.* (1984)).

In the continuum equations of fluid mechanics for normal density gases, the boundary conditions are usually given as:

1. The flow velocity at the wall is set equal to the wall velocity (the no-slip condition).
2. The temperature of the system at the wall is set equal to the wall temperature.
3. The density of the system at the wall is assumed to be unaffected by the presence of the wall.

However, in the case of granular systems, these simple conditions can no longer be used:

1. Experimental evidence indicates that slip between the flow velocity and the wall velocity is a common feature of granular flow problems (Hanes (1987)).
2. In the kinetic theory of gases, the wall is often treated as a single, infinitely large, infinitely massive particle (Henderson *et al.* (1976), Waisman *et al.* (1976)). If we apply this model to the case of granular flow, then the second moment of the wall's velocity may be regarded as the wall's "temperature." This temperature will be determined by the balance between three effects: the generation of thermal energy by the shear stress and slip velocity, the conduction of thermal energy between the wall and the particles, and the loss

of thermal energy due to the inelasticity of grain-wall collisions. Since these mechanisms differ in details from the corresponding mechanisms in the bulk of the flow, the temperature of the wall may be significantly different from the temperature of the particles at the wall.

3. These differences between grain-wall and grain-grain collisions may also cause the bulk density of grains at the wall to differ from the bulk density away from the wall. If, for example, the rebound velocity of a particle in a grain-wall collision were lower on average than the corresponding rebound velocity in a grain-grain collision, then each grain at the wall would occupy a smaller volume of space than a grain in the bulk would occupy. Thus the bulk density (or equivalently the number density) of grains at the wall would be higher than the density away from the wall.

The first two of these effects have been incorporated in previous formulations of the boundary conditions.

Hui *et al.* (1984) presented a set of boundary conditions for the phenomenological theory of Haff (1983) based upon the rates of energy and momentum transfer at a wall. Although the slip velocity of the grains at the wall was included in the calculation of the momentum transfer, the thermal energy which would be generated by this slip was not included in the derivation of the energy transfer. Thus, the case in which a significant portion of the internal thermal energy of a granular flow is supplied by slip at the wall cannot be treated within this framework.

Another set of boundary condition equations has been given by Jenkins and Richman (1986), both for a two-dimensional system of smooth circular disks as well as a three-dimensional system of spheres. Employing methods of averaging from the kinetic theory of dense gases, they derive expressions for the rate at which linear momentum and energy are transferred between the granular flow and the wall. Equating these expressions to the corresponding rates in the flow gives the boundary conditions. These authors were the first to emphasize the role of the normal stress boundary condition. However, in the applications they discuss, a "slip" in number density at the wall was not allowed for, leading to an over-constrained set

of equations *i.e.*, an additional boundary condition had been introduced, on the pressure, but no additional variables. Thus, the solution of a steady state Couette flow problem required a unique number of flow disks across the gap. The physical difficulty with this result can be seen in the limit of zero wall velocity, where the Couette flow problem reduces to the problem of particles in a box with no flow. In the absence of allowing for a density "slip" at the wall, there is in general no way to arrive at a steady-state population of particles in the box.

The theory of boundary conditions for three-dimensional systems presented here is derived in a manner similar to that used by Haff (1983) in obtaining the equations of motion for the bulk flow of smooth particles and to that used in Chapter 1 in obtaining the equations of motion for the bulk flow of rough, spinning particles. In this model each microscopic process of interest such as momentum transfer in grain-grain collisions, momentum transfer in grain-wall collisions, energy absorption in grain-grain collisions, and so forth, is considered explicitly, and the corresponding local expressions for energy and momentum transfer within the bulk and between the bulk and the walls are derived. These expressions are suitably averaged and combined in order to arrive at the desired equations of motion, constitutive relations, and boundary conditions. This approach does not start with a particle distribution function, contrary to the tack taken in some applications of kinetic theory, and hence it cannot calculate the precise magnitude of the dimensionless coefficients ($q$, $r$, $t$, *etc.*; see below) which characterize each physical process when those processes are combined together in a balance law. On the other hand, in the present model these factors are not arbitrary but are known to be of order unity. Jackson (1986) has discussed how rigorous kinetic theory gives results which differ only slightly from ours. The advantage of the heuristic approach used here is that specific physical processes are identified clearly from the start at the microscopic level, and that their role in the equations of motion, constitutive relations and boundary conditions remains clear by virtue of the unique tag they carry in the form of a specific dimensionless constant. These constants are not intended to be used as "fitting parameters," but rather as indicators of the importance and role

of specific microprocesses. We also note that, over a very wide range in velocities and densities, there may in fact be slight variations in the values of the "constants" of the model (as is also true in kinetic theory). For our purpose here, which is to outline some new and interesting boundary effects in granular systems, we neglect any such variation, which is expected to be small.

Working within this framework, our approach will be to introduce "slips" in the bulk density and thermal velocity, the necessity for which is argued below, as well as the more conventional slip in average velocity, and then to solve the problem of Couette flow, in which the properties of the grains and wall along with the wall velocity and Couette-cell width are given, and the pressure, shear stress, and velocity profiles in the bulk are calculated.

It should be noted that in the kinetic theory of granular flow presented by Haff (1983), the bulk density $\rho$ is assumed to be essentially constant throughout the flow. This assumption is adopted here, so that the results apply mainly to dense systems. Where small variations in the bulk density would have a significant effect (as in the collision rate), the variations are allowed by the use of the grain-to-grain spacing variable $s$. We will continue to use this formalism in describing the variation in density at a wall.

### The Grain-Wall Collision Model

The grains are assumed to be identical, inelastic, smooth spheres of diameter $d$ and mass $m$. (In order to simplify our treatment of the boundary conditions, the spin of the grains will be ignored.) The packing fraction is taken to be high (*i.e.*, $s < d$) and the system is taken to be sufficiently agitated that the particles undergo only binary collisions so that the theory of Haff (1983) can be applied. In a collision, the particle is assumed to contact a section of the wall which has a local unit normal vector $\vec{k}$ (figure 2.1) and a coefficient of restitution $e_w$.

In calculating the results of a grain-wall collision, the mass of the wall will be taken as infinitely greater than the mass of a grain.

The following assumptions are made about wall roughness:

1. At the microscopic level (smaller than a grain diameter), the wall is smooth and frictionless.
2. On a scale slightly larger than a grain diameter, the surface of the wall has a shape consisting of smooth undulations of amplitude less than a grain diameter. The assumption of a small amplitude is based on the idea that any feature of greater amplitude will tend to trap one or more particles, thus "healing" itself and forming a new boundary with small amplitude undulations. (Surface "healing" is commonly seen in computer simulations of flow, Haff (1987); see also discussion of self-bounding fluids in Hui *et al.* (1984).) The surface roughness is characterized by a distribution function for $\vec{k}$, $f(\vec{k})$, which is assumed to be isotropic.
3. On average over distances much larger than a grain diameter, the wall is flat.

In the present treatment of the boundary conditions, the position of the wall as a function of time will be allowed to have a random component. This random component will have an amplitude less than a grain diameter and can vary on a time scale similar to the time between collisions of the grains in the flow. (Motion with larger amplitudes or on longer time scales should be included in the macroscopic description of the boundary's position.) This effective random motion can arise from two uncorrelated sources: the vibrational motion of the wall and the slip velocity.

The vibrational motion of the wall will be described by the second moment of its velocity, designated $v_w$. Even though this motion will be correlated over the entire length of the wall, it will be treated as contributing an uncorrelated random motion to the grains in view of the fact that the positions of the grains are not correlated over distances greater than a few grain diameters. Further discussion of the analogy between a small-amplitude high-frequency wall vibration and a thermal source is given by Haff (1983).

The random motion due to the slip velocity can be quantified by considering the frame of reference in which the average velocity of the granular flow near the wall is zero. In this frame, the slip velocity coupled with the wall's roughness results in a fluctuation in the normal component of velocity in a grain-wall collision. This

component of the surface velocity in the $\vec{k}$ direction (figure 2.1) is $\vec{u}_s \cdot \vec{k}$, where $\vec{u}_s$ is the slip velocity ($\vec{u}_s$ is taken to point along the wall, the $\vec{r}$ direction in figure 2.1). Averaging this dot product over all possible values of $\vec{k}$, and adding it in quadrature to any average externally driven vibrational velocity, gives the wall's effective vibrational velocity:

$$v_{w,\ eff}^2 = v_w^2 + \int (\vec{u}_s \cdot \vec{k})^2 f(\vec{k}) d\vec{k} = v_w^2 + u_s^2 \int (\vec{k} \cdot \vec{r})^2 f(\vec{k}) d\vec{k} \equiv v_w^2 + u_s^2 < k_r^2 > . \quad (2.1)$$

In this equation, we have explicitly quantified the roughness of the wall in the term $< k_r^2 >$. A perfectly flat wall will have $< k_r^2 >= 0$; and an increasingly rougher wall will have increasing values of $< k_r^2 >$.

Finally, the average rate at which grain-wall collisions take place is given by the relative velocity of the grain and the wall divided by the average grain-wall spacing:

$$\frac{(v^2 + v_w^2 + u_s^2 < k_r^2 >)^{\frac{1}{2}}}{s_w}, \quad (2.2)$$

where $v$ is the average thermal velocity of the grains.

Since these sources of vibrational motion are regarded as random and uncorrelated, they are added in quadrature and the time average of any cross terms is assumed to vanish.

## The Boundary Condition Equations

*The pressure*

The pressure on the wall can be found by means of the cell model (Hirschfelder, Curtiss, and Bird (1964)). The normal component (along $\vec{n}$ in figure 2.1) of the momentum transferred to the wall in a single grain-wall collision is of order

$$m(v^2 + v_w^2 + u_s^2 < k_r^2 >)^{\frac{1}{2}}.$$

Since the particle occupies a cell whose dimensions are of order $d$, the area across which this momentum transfer takes place is approximately $d^2$. Combining these

values with the rate at which grain-wall collisions take place (equation 2.2) gives the pressure on the wall:

$$p = t_w d\rho \frac{(v^2 + v_w^2 + u_s^2 < k_\tau^2 >)}{s_w}, \tag{2.3}$$

where all of the proportionality constants have been incorporated into $t_w$, a dimensionless constant of order 1.

The pressure in the granular flow (as given by Haff) is $p = td\rho v^2/s$, where $t$ is a dimensionless constant of order 1. Setting these two expressions equal gives the value of the grain-wall spacing:

$$s_w = \frac{t_w}{t}\left(\frac{v^2 + v_w^2 + u_s^2 < k_\tau^2 >}{v^2}\right) s. \tag{2.4}$$

This boundary condition is the result of the fact that a particle in the layer adjacent to the wall "sees" a different environment on one side of its cell from the others. In order to transmit a constant pressure in the direction perpendicular to the wall, the grain-wall spacing must adjust accordingly. This is equivalent to a "slip" or jump in the bulk density of the system at the wall. (Although density and mean free path do not stand in a strictly one-to-one relation at high density, because of geometrical packing effects, we equate for the purposes of this paper, "density slip" and "mean free path slip," see Haff (1983).)

This "slip" in the bulk density is actually a first order approximation to the more complex oscillations in bulk density seen in calculations and simulations of hard-sphere fluids bounded by a flat wall (Henderson *et al.* (1976), Snook and Henderson (1978), Waisman *et al.* (1976)). These variations in bulk density can arise even though the particle-particle and particle-wall collisions are perfectly elastic, simply due to the layering effect of the particles near a flat wall (Snook and Henderson (1978)). Since the wall we use here is not perfectly flat, it would not be appropriate to go beyond this first-order approximation.

*The shear stress*

On the average, a collision between a grain and a rough wall will have a component of momentum transfer along the direction of the slip velocity. This momentum

transfer results in the transmission of shear stress between the wall and the granular flow.

To calculate this shear stress, note that the normal velocity in a grain-wall collision due to the slip velocity is $(\vec{u}_s \cdot \vec{k})\vec{k}$; and its component parallel to the wall is $(\vec{u}_s \cdot \vec{k})(\vec{k} \cdot \vec{\tau}) = u_s k_\tau^2$. Multiplying this by the particle mass and averaging over all possible values of $\vec{k}$ gives the average component of momentum transfer parallel to the wall:

$$mu_s \int k_\tau^2 f(\vec{k}) d\vec{k} \equiv mu_s < k_\tau^2 > .$$

Taking into account the area and rate of collisions, the flux of lateral momentum will be

$$\sigma = q_w d\rho u_s < k_\tau^2 > \frac{(v^2 + v_w^2 + u_s^2 < k_\tau^2 >)^{\frac{1}{2}}}{s_w}, \tag{2.5}$$

where $q_w$ is a dimensionless constant of order 1.

Equation (2.5) seems to make a puzzling prediction, namely, that the shear stress vanishes if the slip velocity $u_s$ is zero. Yet we know that, for fluids in general, a no-slip boundary condition does *not* imply a vanishing stress. In the microscopic granular flow model of Haff (1983) used here, $u_s$ is a dependent variable, a quantity whose value must be computed. It is not something we can adjust by hand. Therefore, the no-slip condition is not to be specified a priori by setting $u_s = 0$, but is a condition which might or might not turn out to have validity in the course of the calculation. And, in particular, the no-slip condition does not mean $u_s = 0$. It only means that $u_s$ is small compared with the total shear $U$ (we set $v_w = 0$ for simplicity). In fact, $u_s$ cannot vanish under any flow conditions save the no-flow case as can be seen by noting that all velocities are scaled by $U$.

The shear stress in the granular flow (Haff (1983)) is

$$\sigma = \eta \frac{\partial u}{\partial y} = qd^2 \rho \frac{v}{s} \frac{\partial u}{\partial y},$$

where $q$ is a dimensionless constant of order 1.

Equating these two fluxes of lateral momentum gives the boundary condition relating the slip velocity and the normal derivative of the flow velocity:

$$\frac{\partial u}{\partial y} = \frac{q_w}{q} \frac{u_s}{d} < k_\tau^2 > \frac{(v^2 + v_w^2 + u_s^2 < k_\tau^2 >)^{\frac{1}{2}}}{v} \frac{s}{s_w}.$$

Substituting from equation 2.4 for the ratio of spacings gives

$$\frac{\partial u}{\partial y} = \frac{q_w}{q}\frac{t}{t_w}\frac{u_s}{d} < k_\tau^2 > \frac{v}{(v^2 + v_w^2 + u_s^2 < k_\tau^2 >)^{\frac{1}{2}}}. \tag{2.6}$$

This nonlinear relation between the slip velocity and the shear rate reduces to that obtained by Hui (1984) in the limit where the particle fluctuation velocity is much greater than the wall's effective fluctuation velocity. Again, for the same reasons as in equation (2.5), the flow velocity gradient remains non-zero even when the no-slip approximation is a good one.

The condition of isotropy in the distribution function $f(\vec{k})$ allows us to assume collinearity of the shear stress and slip velocity at the wall. If $f(\vec{k})$ were anisotropic (*e.g.*, a "washboard" wall surface oriented obliquely to the flow), the shear stress would be related to the slip velocity through a second rank tensor.

*The thermal energy flux*

The thermal energy of particles colliding with a wall is affected by two competing processes. The inelasticity of grain-wall collisions results in a loss of thermal energy, while the effective fluctuation velocity of the wall will supply thermal energy to the grains.

The loss of thermal energy in a collision due to the particle's thermal velocity and inelasticity will be of order

$$m(1 - e_w^2)v^2.$$

The gain of thermal energy in a collision due to the wall's effective temperature will be of order

$$m(1 + e_w)^2 v_{w,\,eff}^2.$$

Combining these two effects, substituting for the wall's effective temperature (equation 2.1) and factoring in the area and rate of collisions, we get an expression for the flux of thermal energy at the wall:

$$Q = -r_w d\rho \left[ v_w^2 + u_s^2 < k_\tau^2 > - \frac{(1 - e_w)}{(1 + e_w)} v^2 \right] \frac{(v^2 + v_w^2 + u_s^2 < k_\tau^2 >)^{\frac{1}{2}}}{s_w}. \tag{2.7}$$

The flux normal to the wall of thermal energy in the granular flow (Haff (1983)) is

$$Q = -K\frac{\partial}{\partial y}\left(\frac{\rho v^2}{2}\right) = -rd^2\frac{v}{s}\rho\frac{\partial}{\partial y}(\frac{1}{2}v^2),$$

where $r$ and $r_w$ are dimensionless constants of order 1.

Setting these expressions equal and substituting equation 2.4 gives the final boundary condition

$$\frac{\partial}{\partial y}(\frac{1}{2}v^2) = \frac{r_w}{r}\frac{t}{t_w}\frac{1}{d}\left[v_w^2 + u_s^2 < k_\tau^2 > -\frac{(1-e_w)}{(1+e_w)}v^2\right]\frac{v}{(v^2 + v_w^2 + u_s^2 < k_\tau^2 >)^{\frac{1}{2}}}. \tag{2.8}$$

### Steady State Couette Flow

In gravity-free, steady state Couette flow, the walls are driven at a given velocity parallel to their surfaces. Figure 2.2 illustrates the geometry of the system; the plates are of infinite extent in the $x$ and $z$ directions with the origin of the $y$-axis midway between them. The number of grains in the channel will be specified by the parameter $\Delta h$, the free space remaining when all of the grains are packed towards one wall (figure 2.3). Due to symmetry the flow variables will be functions of $y$ only. Combining this with the condition $\vec{\nabla}\cdot\vec{u} = 0$ leads to the conclusion that the only non-zero component of the flow velocity $\vec{u}$ is $u_x$.

The equation governing the evolution of momentum in a steady granular flow is (Haff (1983))

$$\frac{\partial}{\partial t}(\rho u_i) = \frac{-\partial}{\partial x_k}\left[p\delta_{ik} + \rho u_i u_k - \eta\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right)\right].$$

Evaluating the $x$ and $y$ components of this equation gives the respective results that the shear stress $\left(\sigma_0 = \eta\frac{\partial u_x}{\partial y}\right)$ and the pressure $(p_0)$ are constant throughout the flow.

The equation for the total kinetic and thermal energy of the system is

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\rho u^2 + \frac{1}{2}\rho v^2\right)$$

$$= \frac{-\partial}{\partial x_k}\left[\rho u_k\left(\frac{p}{\rho} + \frac{1}{2}u^2 + \frac{1}{2}v^2\right) - u_i\eta\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right) - K\frac{\partial}{\partial x_k}\left(\frac{1}{2}\rho v^2\right)\right] - I.$$

Setting to zero the derivatives with respect to $t$, $x$, and $z$, and setting $u_y = u_z = 0$ leaves

$$0 = \frac{-\partial}{\partial y}\left[u_x \eta \frac{\partial u_x}{\partial y} - K\frac{\partial}{\partial y}\left(\frac{1}{2}\rho v^2\right)\right] - I.$$

The coefficient of thermal diffusivity ($K$) and the thermal energy sink ($I$) are given by

$$K = rd^2\frac{v}{s}, \qquad I = \gamma\rho\frac{v^3}{s},$$

where $\gamma$ is a dimensionless constant related to the coefficient of restitution of a particle-particle collision $e$ ($\gamma$ is proportional to $(1-e^2)$).

Substituting for these and for the constant shear stress and pressure gives the equation for the thermal velocity within the granular flow

$$0 = \frac{\partial^2 v}{\partial y^2} + \omega^2 v, \tag{2.9}$$

$$\text{where } \omega^2 = \frac{1}{rd^2}\left(\frac{t^2\sigma_0^2}{qp_0^2} - \gamma\right). \tag{2.10}$$

The general solution for the thermal velocity is

$$v(y) = 2v_0\cos(\omega y). \tag{2.11}$$

Both $v_0$ and $\omega$ (*i.e.*, $\sigma_0/\rho_0$) are determined by the boundary conditions. (Note that this assumes $\omega^2 > 0$; for the case of $\omega^2 < 0$, the trigonometric functions in this and succeeding equations are replaced by the corresponding hyperbolic functions.)

Combining equation 2.3 (with pressure equal to $p_0$) and equation 2.5 (with shear stress equal to $\sigma_0$), we can solve for the slip velocity in terms of the thermal velocity of the grains at the wall

$$u_s = v(h/2)\frac{\sigma_0 t_w}{p_0 q_w < k_\tau^2 >}\left[1 - \frac{\sigma_0^2 t_w^2}{p_0^2 q_w^2 < k_\tau^2 >}\right]^{-\frac{1}{2}}. \tag{2.12}$$

Substituting this into equation 2.8 gives

$$\omega d D\left[\tan\left(\frac{\omega h}{2}\right)\right] = \left[E - \left(\frac{F\sigma_0^2/p_0^2}{1 - F\sigma_0^2/p_0^2}\right)\right]\left(1 - \frac{F\sigma_0^2}{p_0^2}\right)^{\frac{1}{2}}, \tag{2.13}$$

where $D = \frac{rt_w}{r_w t}$, $E = \frac{1-e_w}{1+e_w}$, and $F = \frac{t_w^2}{q_w^2 < k_r^2 >}$. This is a transcendental equation for $\sigma_0^2/p_0^2$ in terms of the particle properties ($d$, $q$, $r$, $t$, and $\gamma$), the wall properties ($e_w$, $q_w$, $r_w$, $t_w$, and $< k_r^2 >$) and the inter-wall spacing $h$. The equation can be solved numerically to obtain the shear stress to pressure ratio as a function of wall roughness $< k_r^2 >$. An example is given in figure 2.4. This ratio represents the tangent of the effective dynamical internal friction angle of the grain-mass-plus-wall system, which is essentially zero for nearly smooth walls, since the walls offer almost no resistance to shear. As the walls are roughened, the coupling between the walls and the grain mass increases (as evidenced by the rapidly decreasing slip velocity (figure 2.5)), finally reaching a point of saturation beyond which increasing wall roughness has little effect. This presumably reflects the fact that the main grain mass has become much weaker to shear than the wall-grain layer. This is a significant result because it means that if the walls of a shear cell apparatus designed to measure internal stresses are insufficiently rough, the measurements relate principally not to the effective internal friction in the bulk, but to the effective friction offered by the wall.

The average flow velocity is obtained from integrating $du/dy = \sigma_0/\eta$:

$$u(y) = \frac{2t}{q}\frac{\sigma_0}{p_0}\frac{v_0}{\omega d}\sin(\omega y), \tag{2.14}$$

where now $u \equiv u_x$.

The wall velocity (with respect to the center of the channel where the flow velocity vanishes) is the sum of the slip-velocity and the flow velocity at the wall:

$$u_w = 2v_0\left[\cos\left(\frac{\omega h}{2}\right)\left(\frac{F\sigma_0^2/p_0^2}{< k_r^2 > (1 - F\sigma_0^2/p_0^2)}\right)^{\frac{1}{2}} + \frac{t}{q}\frac{\sigma_0}{p_0}\frac{1}{\omega d}\sin\left(\frac{\omega h}{2}\right)\right]. \tag{2.15}$$

The ratio of slip velocity to wall velocity is independent of $v_0$, and is shown in figure 2.5. The slip velocity decreases with increasing surface roughness as expected.

The ratio of absorbed thermal energy flux to generated thermal energy flux at the wall is also independent of $v_0$, and is given by

$$\frac{Q_a}{Q_g} = \frac{E(1 - F\sigma_0^2/p_0^2)}{F\sigma_0^2/p_0^2}.$$

This ratio is plotted versus wall roughness in figure 2.6.

The value of $v_0$ is determined by the number of grains in the channel as expressed by the parameter $\Delta h$. Beginning with the expression for $\Delta h$ in terms of $s(y)$:

$$\Delta h = \frac{3}{d}\int_{-h/2}^{-h/2} s(y)dy,$$

substituting $s(y) = td\rho v^2(y)/p_0$, and integrating gives

$$v_0^2 = \frac{\omega \Delta h p_0}{6t\rho[\sin(\omega h) + \omega h]}. \tag{2.16}$$

Combining equations 2.11 and 2.15 gives the ratio of particle thermal velocity at the wall to wall velocity, which is shown in figure 2.7. At zero wall roughness no coupling of the wall to the fluid is possible, and the thermal velocity then vanishes. With increasing wall roughness more and more energy is transmitted to the grain mass and the thermal velocity near the wall increases. However, the slip velocity, which is the source of thermal energy, steadily decreases with increasing $< k_\tau^2 >$ (figure 2.5), and the plot of thermal velocity versus wall roughness shows a maximum, with the thermal velocity slowly decreasing at high values of the roughness. (Whether a maximum always exists is not clear, since, while $u_s$ decreases, $\sigma$ is increasing with $< k_\tau^2 >$ (figure 2.8), and it is their product which determines the energy generation rate.)

Using equations 2.15 and 2.16 we obtain the normalized shear stress:

$$\frac{\sigma_0}{\rho_p d^2 (2u_w/h)^2} = \frac{\frac{3}{8} t \frac{\sigma_0}{p_0} \frac{h^2}{d\Delta h}\left[\frac{\sin(\omega h)}{\omega d} + \frac{h}{d}\right]}{\left[\cos\left(\frac{\omega h}{2}\right)\left(\frac{F\sigma_0^2/p_0^2}{<k_\tau^2>(1-F\sigma_0^2/p_0^2)}\right)^{\frac{1}{2}} + \frac{t}{q}\frac{\sigma_0}{p_0}\frac{1}{\omega d}\sin\left(\frac{\omega h}{2}\right)\right]^2},$$

where $\rho_p$ is the particle material density. Setting $h$ equal to 10 particle diameters and $\Delta h$ equal to 3.25 diameters (corresponding to a volume fraction of 0.5) we obtain plots of the normalized shear stress and normalized pressure as functions of wall roughness (figures 2.8 and 2.9). Both shear stress and pressure vanish for perfectly smooth walls and increase with $< k_\tau^2 >$ as expected.

The variations in flow properties across the channel are shown in figure 2.10, the character of the flow changing as the wall roughness varies. For small values of $< k_\tau^2 >$ (nearly smooth walls) the shear stress to pressure ratio is small (figure 2.4) and $\omega^2 < 0$. In this case, the walls act as a net source of thermal energy flux ($Q_a/Q_g < 1$), and the thermal velocity drops as we move from the wall to the center of the flow (figure 2.10a, curve 1). The flow velocity has a (slight) inflection point (figure 2.10b, curve 1) and the density is greatest in the center of the channel (figure 2.10c, curve 1). As the wall roughness is increased, we may reach the special solution called simple shear flow. Here the thermal velocity (figure 2.10a, curve 2) and the particle-particle separation (figure 2.10c, curve 2) are constant throughout the flow; the flow velocity increases linearly with distance (figure 2.10b, curve 2); and the walls act as neither a source nor a sink of thermal energy flux ($Q_a/Q_g = 1$). For granular systems, simple shear flow is not a common condition; it is only achieved by careful "tuning" of the wall parameters. Finally, for sufficiently rough walls, the shear stress to pressure ratio may be large enough that the rate of thermal energy generation in the flow exceeds the rate of thermal energy loss. This leads to a thermal velocity profile which has a maximum in the center of the flow (*i.e.*, $\omega^2 > 0$, figure 2.10a, curve 3), with the walls acting as a net energy sink ($Q_a/Q_g > 1$). In this case the flow velocity again shows a mid-channel inflection (figure 2.10b, curve 3) while the density is a minimum in the center of the channel (figure 2.10c, curve 3). In general, the shape of the solution to the energy equation as a function of the strength of microscopic processes ($r$, $q$, $t$, *etc.*) can be determined directly from equations (2.10) and (2.13).

Some effects on the thermal velocity of changes in the constants $q$ (the dimensionless constant in the coefficient of viscosity), $t_w$ (the dimensionless constant in the equation of state for particle-wall collisions), and $e_w$ (the coefficient of restitution for particle-wall collisions) are shown in figures 2.11a, b, and c respectively. The particle constants and wall roughnesses are the same as in figure 2.10, except as noted. The curves in figure 2.11 should be compared with those in figure 2.10a. In figure 2.11a, $q$ has been increased by a factor of two. An increase in $q$ means

that a lower shear rate can sustain a higher shear stress, leading to more slip at the wall. This increased slip results in a greater input of thermal energy from the wall to the flow, giving higher thermal velocities at the wall. In figure 2.11b, $t_w$ has been increased by a factor of two. An increase in $t_w$ means the pressure on the wall can be sustained with a larger particle-particle spacing at the wall, again leading to a higher slip velocity. The increase in the flow of thermal energy away from the wall can be seen. In figure 2.11c, $e_w$ has been increased from 0.84 to 0.96. An increase in $e_w$ means that less energy is dissipated in particle-wall collisions. This will also result in a higher flux of thermal energy from the walls to the grains, as shown in the figure. In sum, modest changes of the system constants within this range of expected values lead to modest changes in the associated flow fields.

Finally, we note that throughout most of the range of $< k_\tau^2 >$, the thermal velocity of the particles at the wall is a nearly constant fraction of the wall velocity. This constant fraction decreases as the walls are made more lossy (*i.e.*, as $e_w$ is reduced). The slip velocity as a fraction of wall velocity is seen to decrease with increasing wall roughness, as is expected. However, the lossiness of the walls has little effect on the slip velocity. Thus it appears possible to set the thermal velocity at the wall and the slip velocity there independently by adjusting the lossiness and roughness of the wall.

## Figure Captions

Figure 2.1: Illustration of a particle colliding with a rough wall, showing the unit normal collision vector $\vec{k}$, and the unit vectors normal to the wall ($\vec{n}$) and parallel to the wall ($\vec{\tau}$).

Figure 2.2: Illustration of the Couette flow geometry.

Figure 2.3: Illustration defining the free-space parameter $\Delta h$.

Figure 2.4: Ratio of shear stress to pressure, $\sigma_0/\rho_0$, versus wall roughness, $< k_\tau^2 >$, at fixed wall velocity. The values of the constants used in this numerical simulation are: $\gamma = 0.16$, $q = 0.25$, $r = 1.0$, $t = 1.0$, $e_w = 0.96$ (curve a), $e_w = 0.92$ (curve b), $e_w = 0.84$ (curve c), $q_w = 1.0$, $r_w = 1.0$, and $t_w = 0.5$. No stress is transmitted for perfectly smooth walls. As wall roughness increases, system becomes more resistant to shearing until shear resistance of granular fluid itself becomes determining factor.

Figure 2.5: Ratio of slip velocity to wall velocity, $u_s/u_w$, versus wall roughness, $< k_\tau^2 >$, at fixed wall velocity. The slip velocity decreases with increasing roughness.

Figure 2.6: Ratio of absorbed thermal energy flux at wall to generated energy flux there, $Q_a/Q_g$, versus wall roughness, $< k_\tau^2 >$, at fixed wall velocity.

Figure 2.7: Ratio of particle thermal velocity at the wall to wall velocity, $v(h/2)/u_w$, versus wall roughness, $< k_\tau^2 >$, at fixed wall velocity. The thermal velocity is zero for perfectly smooth walls since in that case no energy can be transmitted from the wall to the fluid. After increasing with increasing roughness, $v/u_w$ falls slightly (in this particular example) because the increase in stress does not quite compensate the decrease in slip velocity, and it is the product of stress and slip velocity which is ultimately responsible for generating heat at the wall.

Figure 2.8: Normalized shear stress, $(\sigma_0/(\rho_p d^2(2u_w/h)^2))$, versus wall roughness, $< k_\tau^2 >$, at fixed wall velocity with $h = 10$ grain diameters and $\Delta h = 3.25$ grain diameters. There can be no transmitted stress for perfectly smooth walls. Stress then increases as $< k_\tau^2 >$ increases.

Figure 2.9: Normalized pressure, $(p_0/(\rho_p d^2(2u_w/h)^2))$, as a function of wall roughness, $< k_\tau^2 >$, at fixed wall velocity. Like shear stress, the pressure must vanish for perfectly smooth walls and increase with increasing $< k_\tau^2 >$.

Figure 2.10: Variations in flow properties across the channel for several wall roughness values. The values of the constants are the same as in figure 2.4, curve c. The wall roughness values are $< k_\tau^2 > = .0625$ (curve 1), $< k_\tau^2 > = .125$ (curve 2), and $< k_\tau^2 > = .25$ (curve 3). Figure 2.10a shows the particle thermal velocity to wall velocity ratio $v/u_w$. Figure 2.10b shows the flow velocity to wall velocity ratio $u/u_w$. Figure 2.10c shows the particle-particle separation to particle diameter ratio $s/d$. See text for discussion.

Figure 2.11: Variations in particle thermal velocity with changes in particle or wall properties. The values of the constants and wall roughness are the same as in figure 2.10, except: $q = 0.5$ in figure 2.11a, $t_w = 1.0$ in figure 2.11b, and $e_w = 0.96$ in figure 2.11c. See text for discussion.

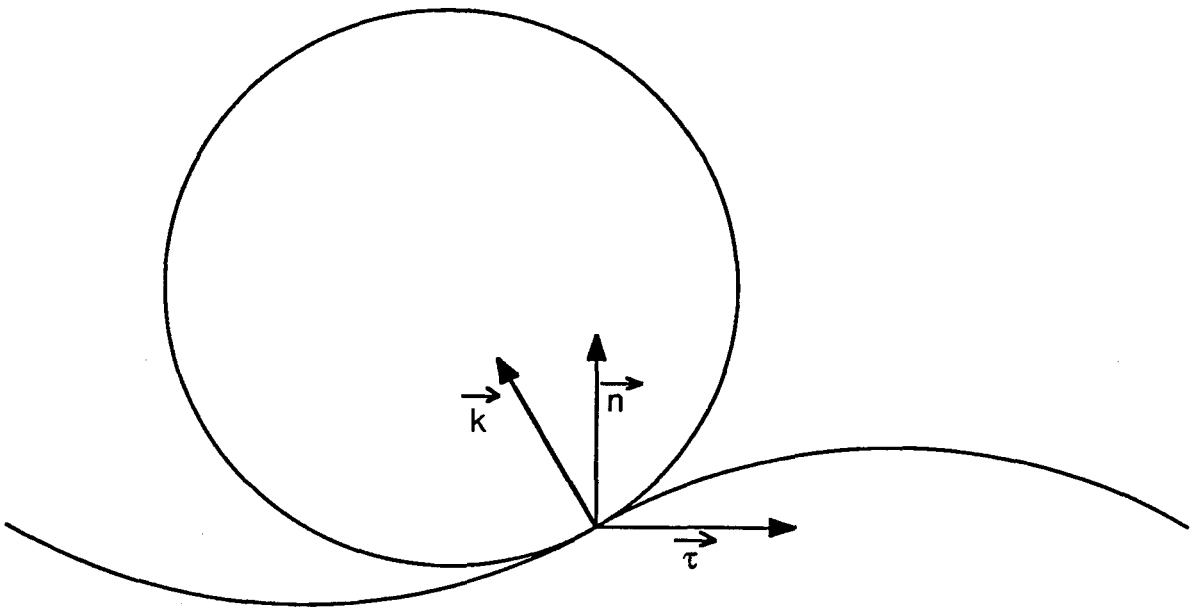



Figure 2.1

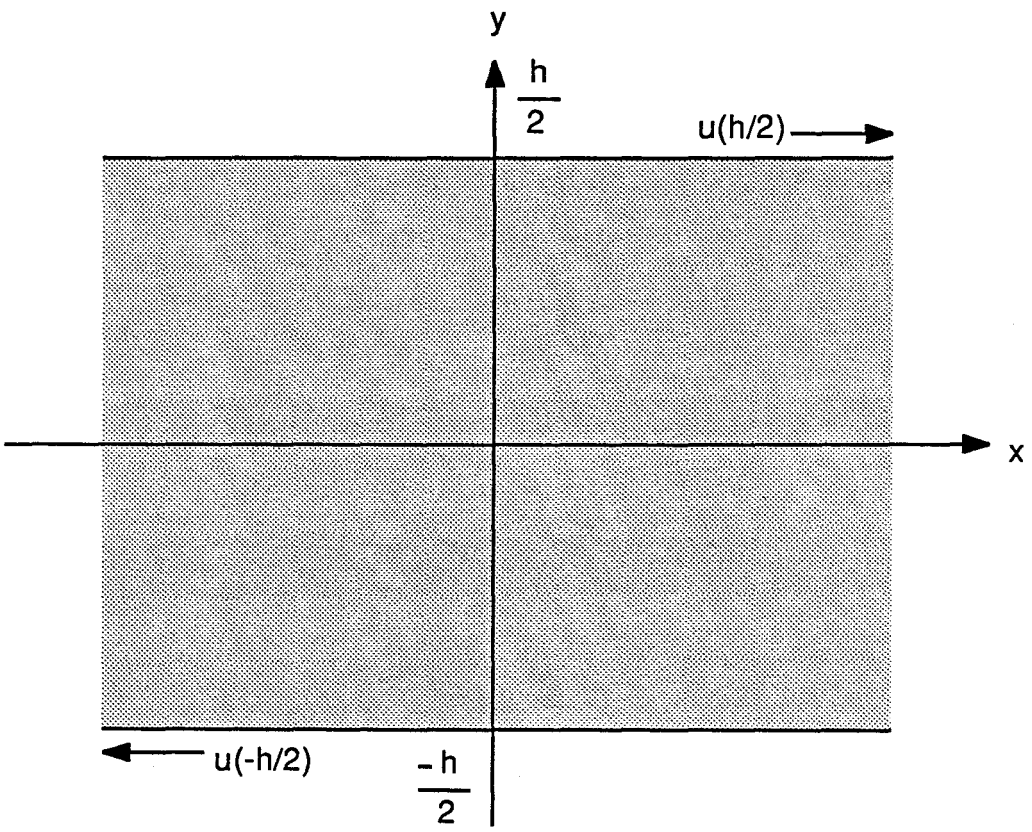



Figure 2.2

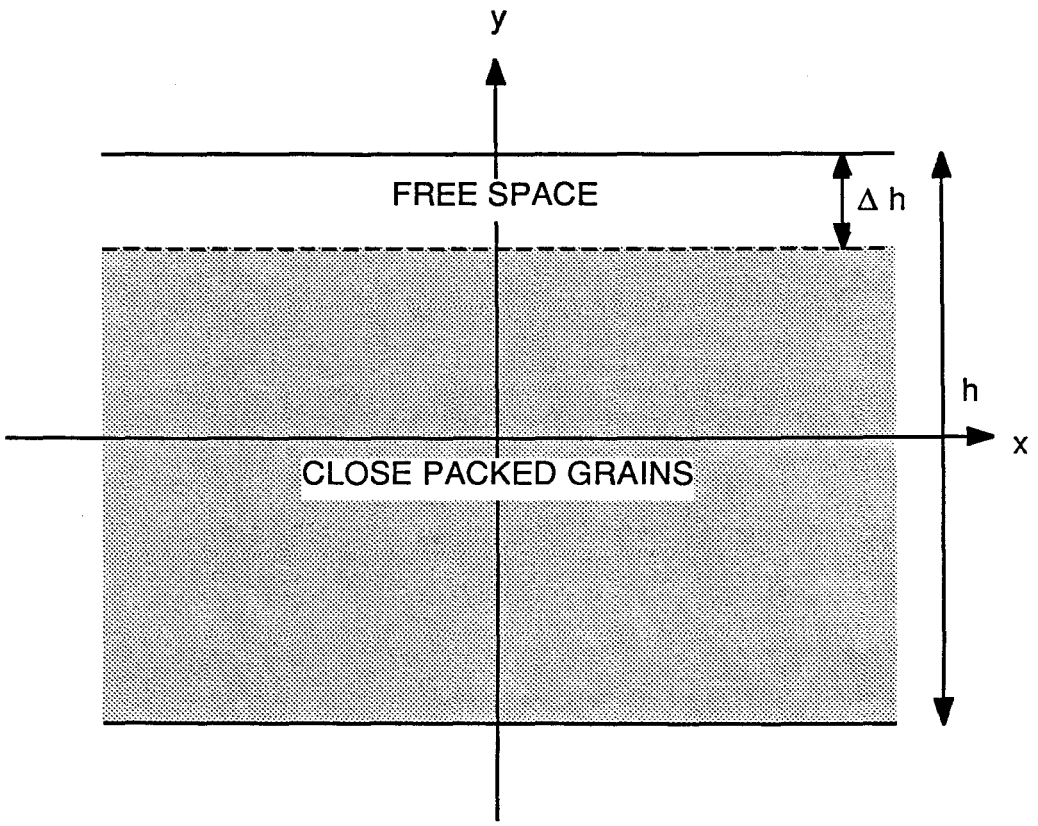



Figure 2.3

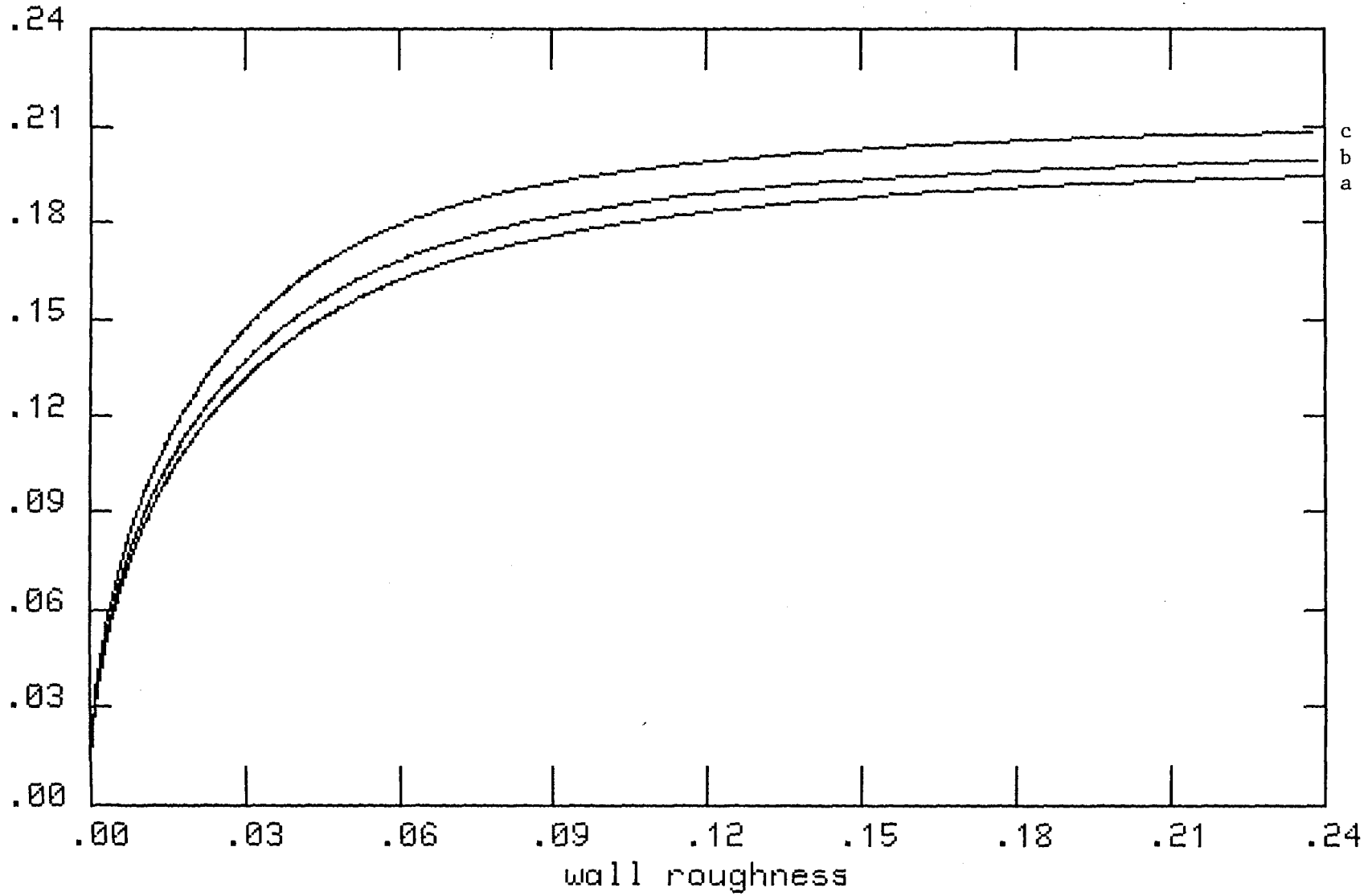



Figure 2.4: Ratio of shear stress to pressure

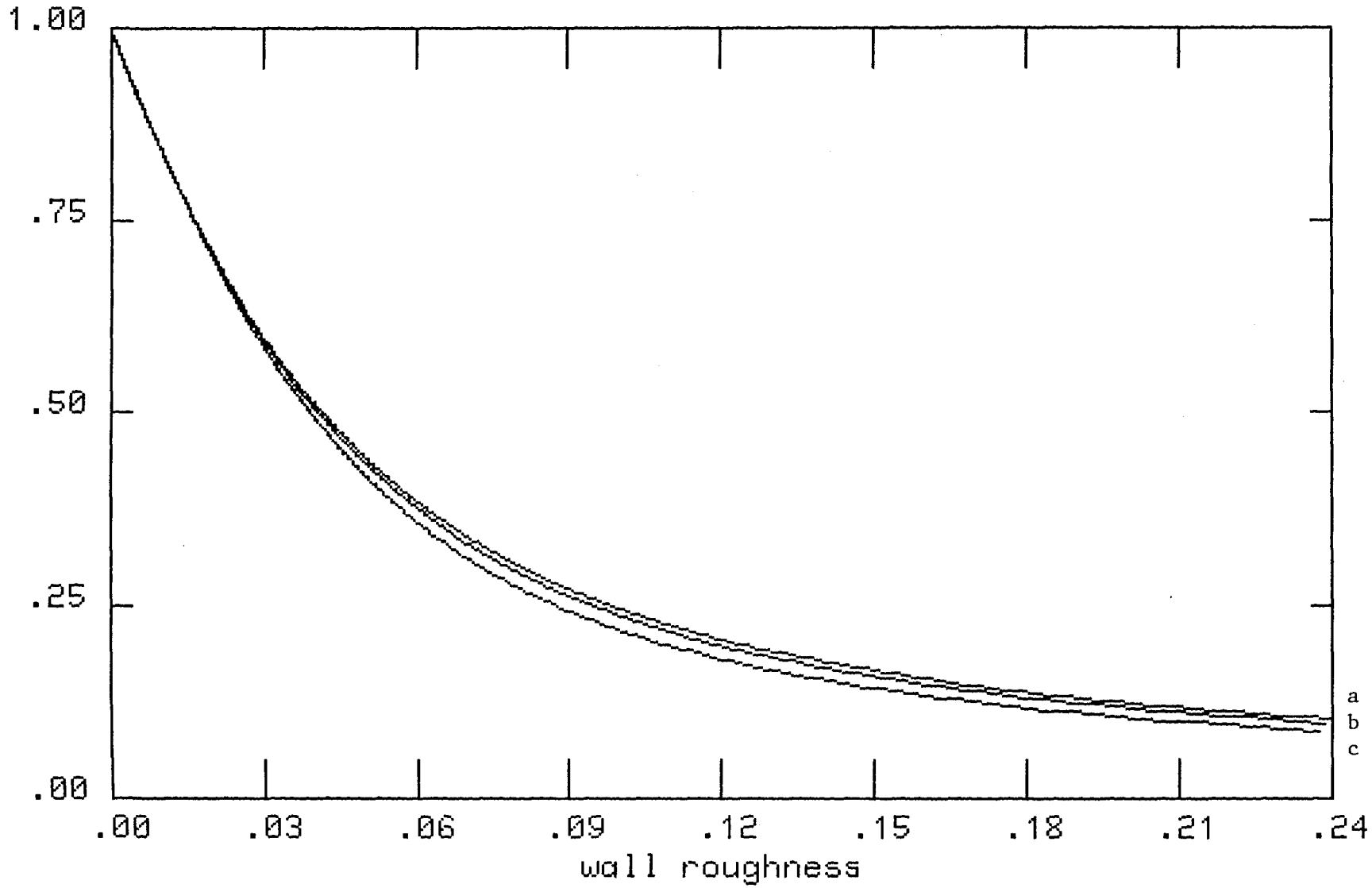



Figure 2.5: Ratio of slip velocity to wall velocity

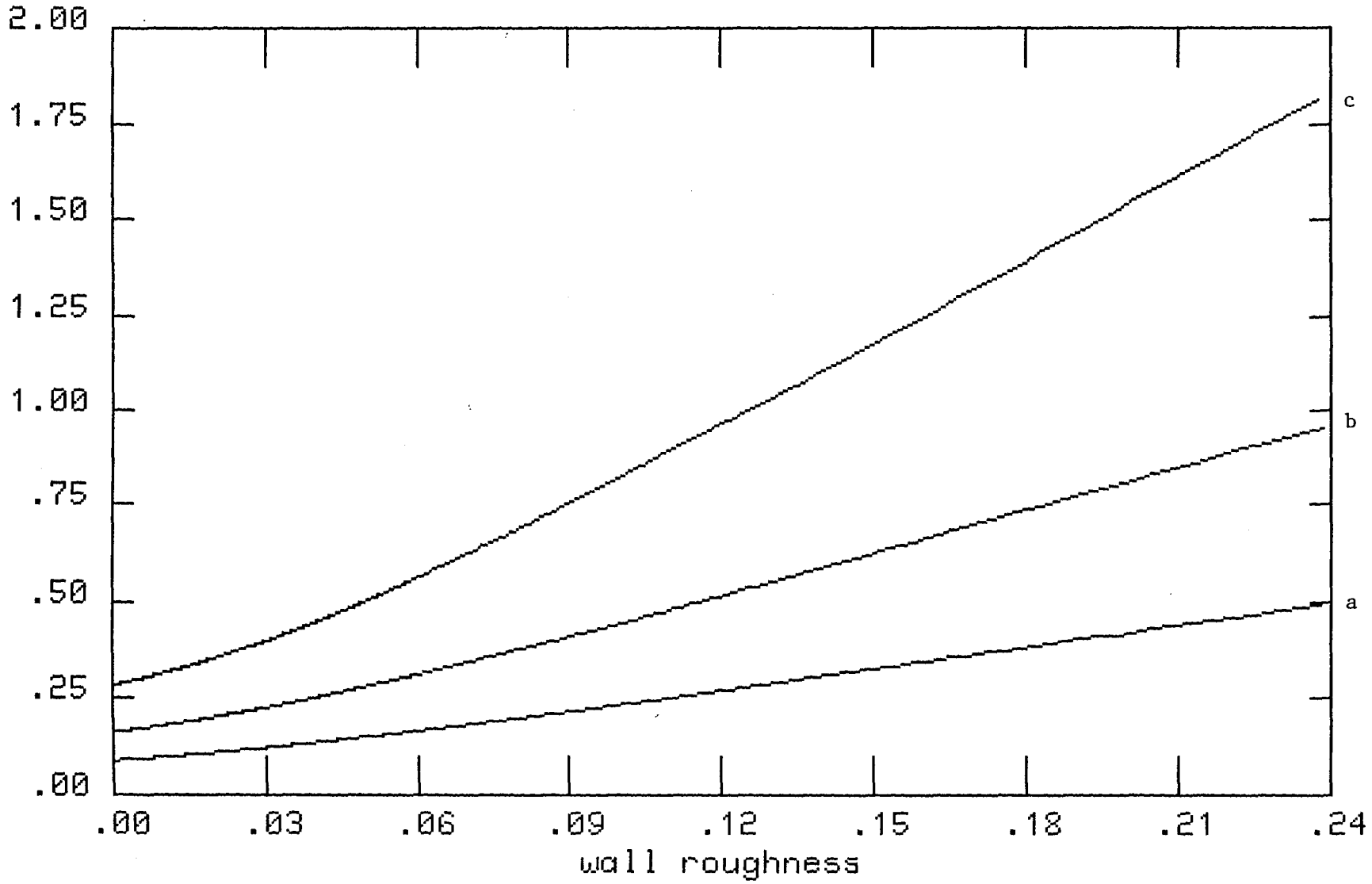



Figure 2.6: Ratio of absorbed thermal energy flux to generated thermal energy flux

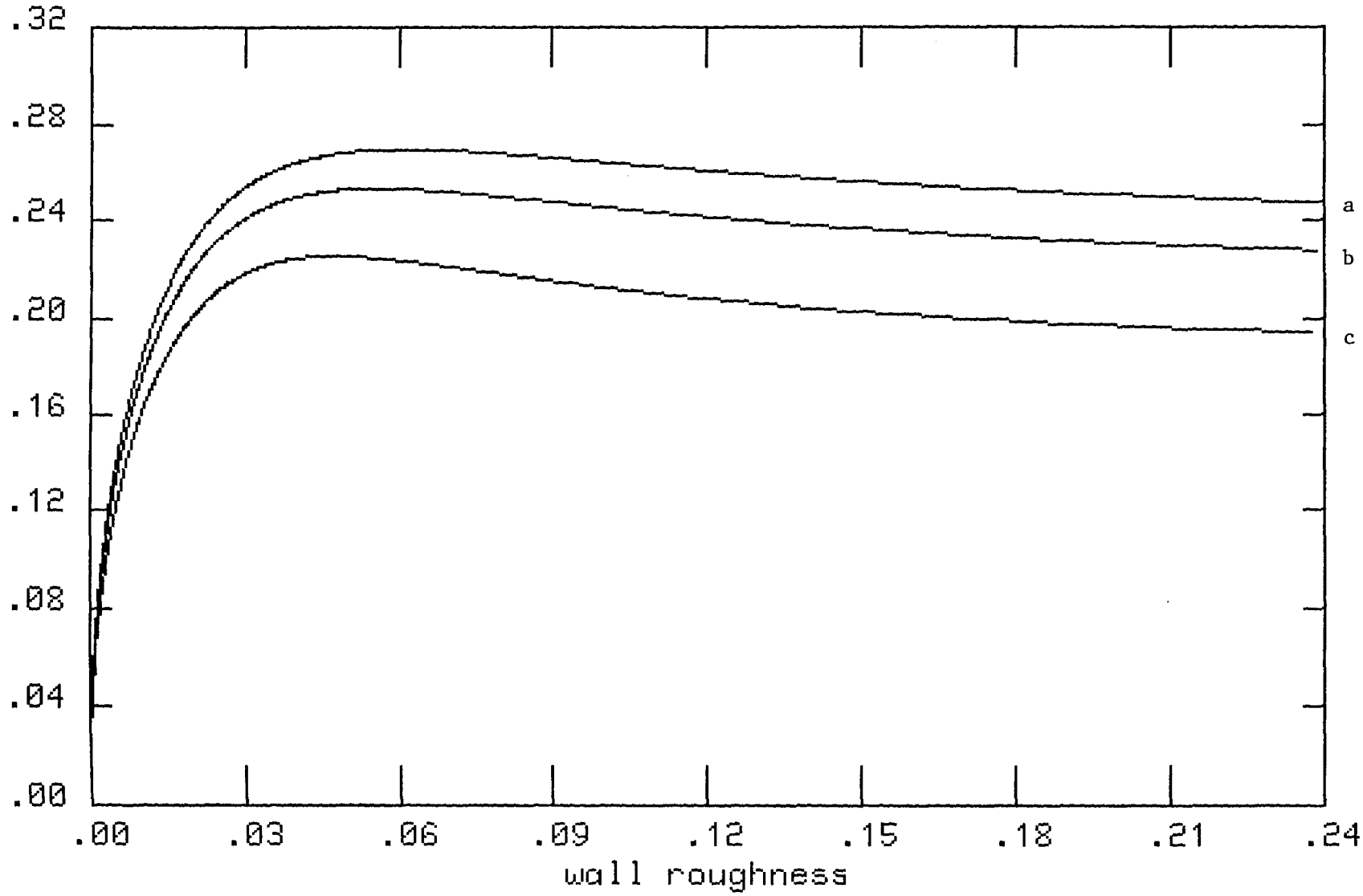



Figure 2.7: Ratio of particle thermal velocity at the wall to wall velocity

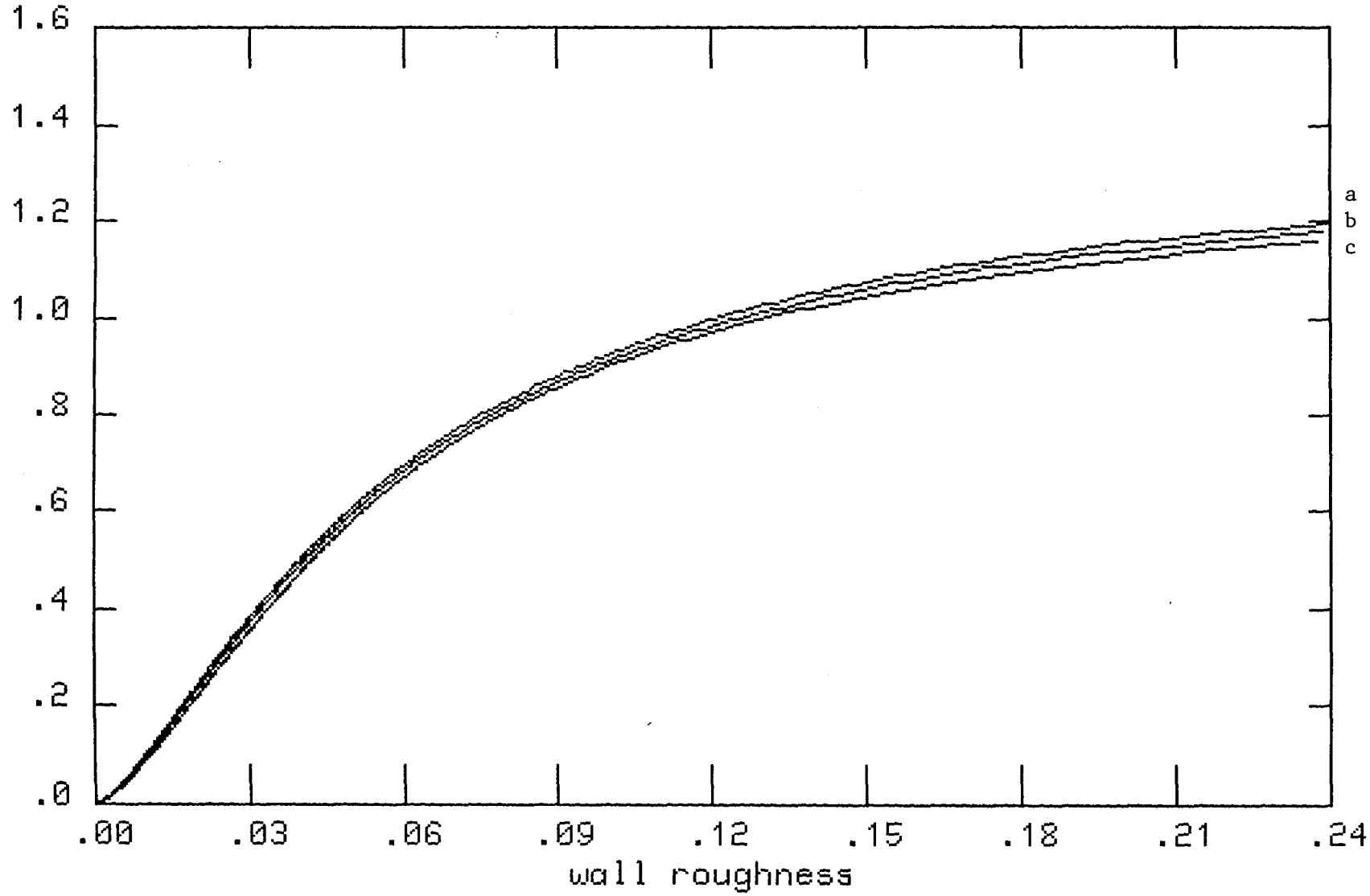



Figure 2.8: Normalized shear stress

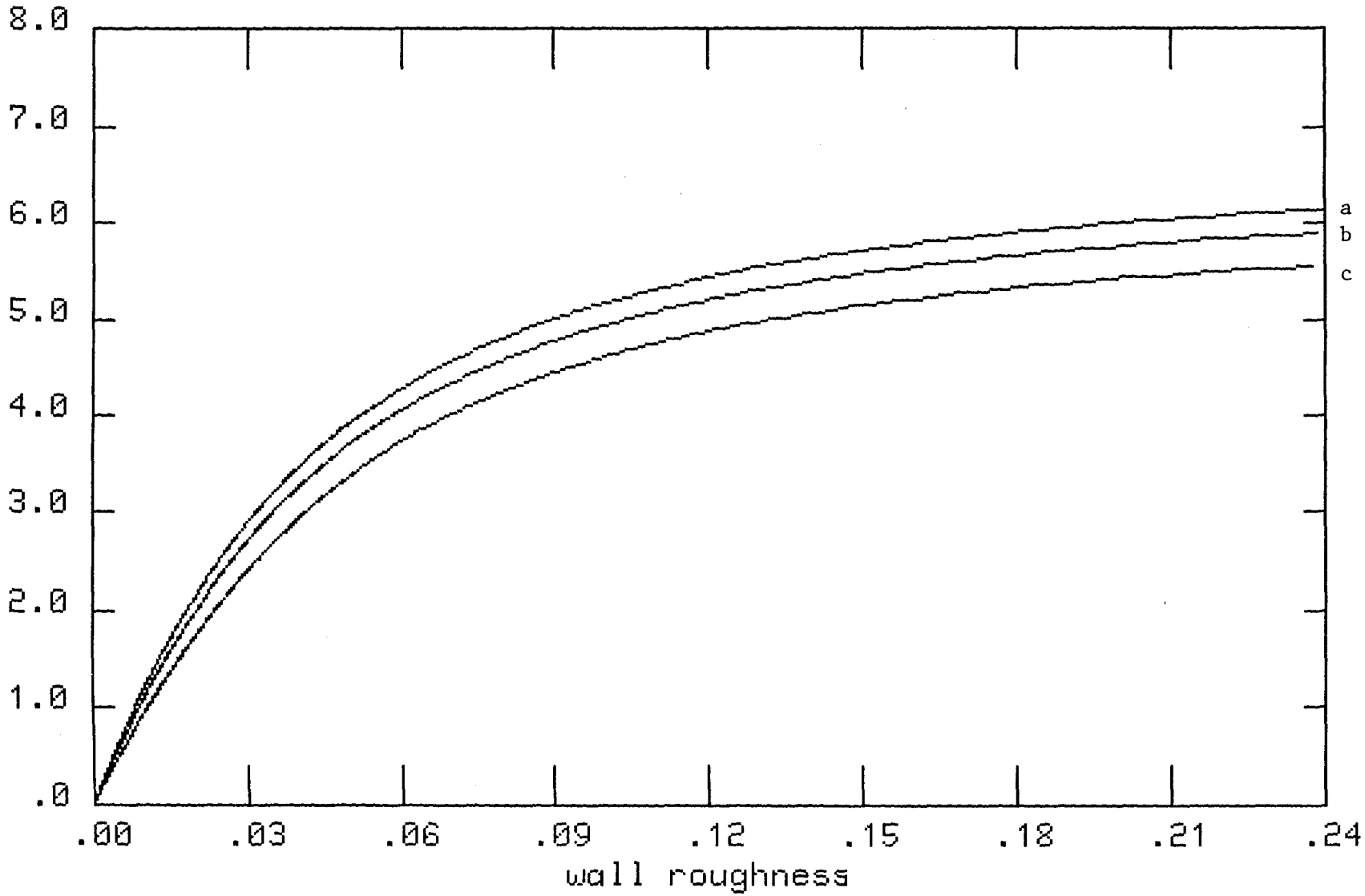



Figure 2.9: Normalized pressure

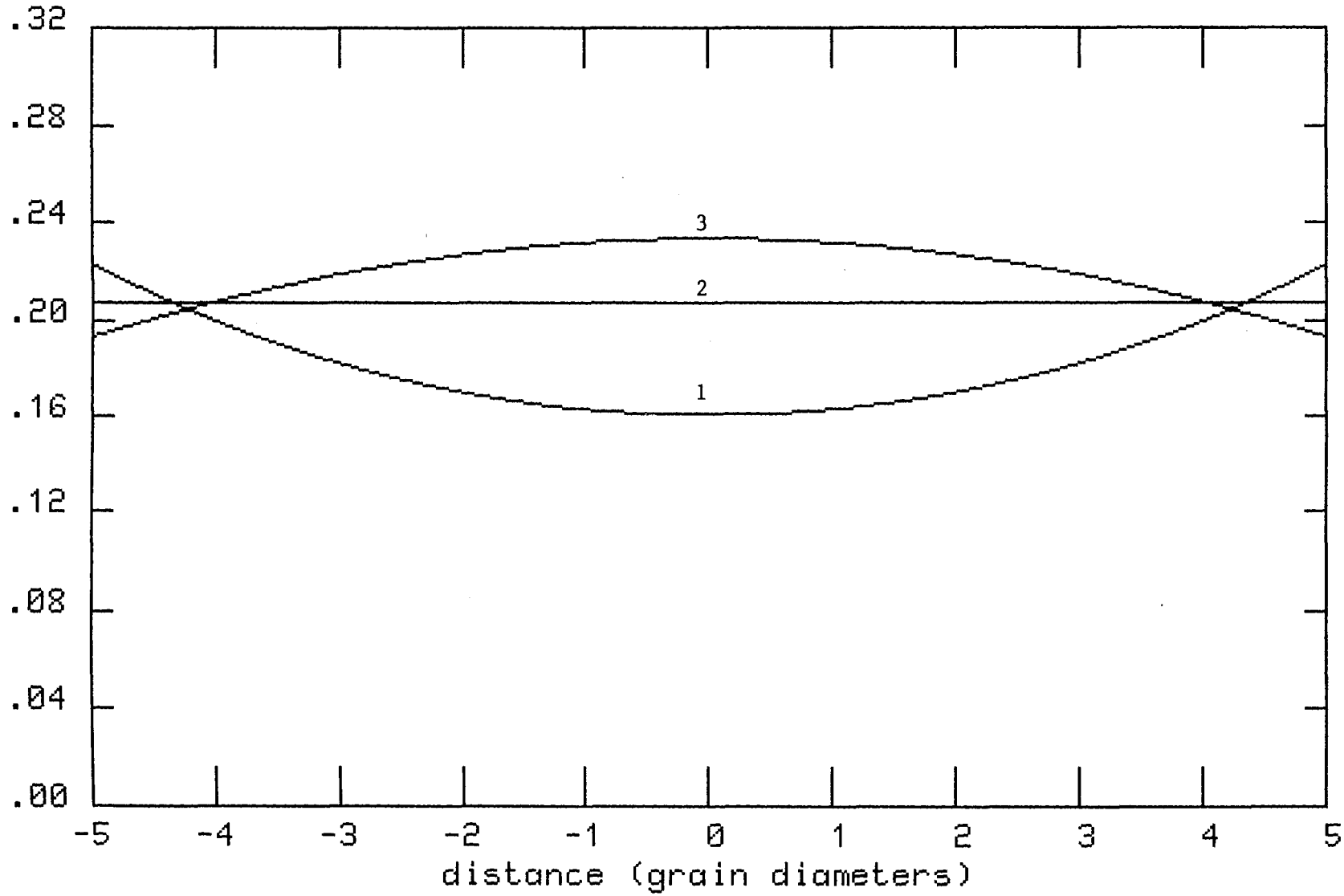



Figure 2.10a: Ratio of particle thermal velocity to wall velocity

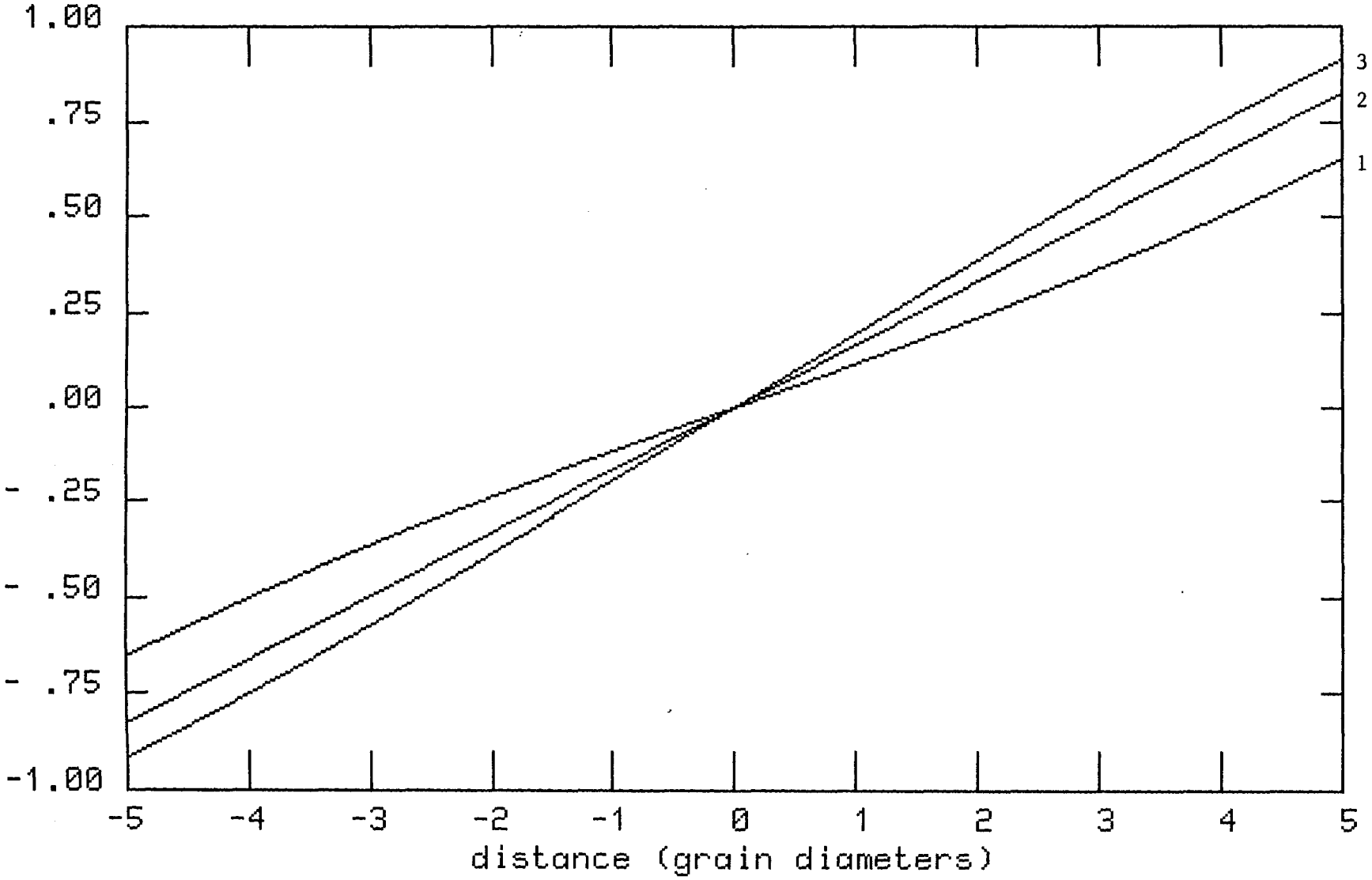



Figure 2.10b: Ratio of flow velocity to wall velocity

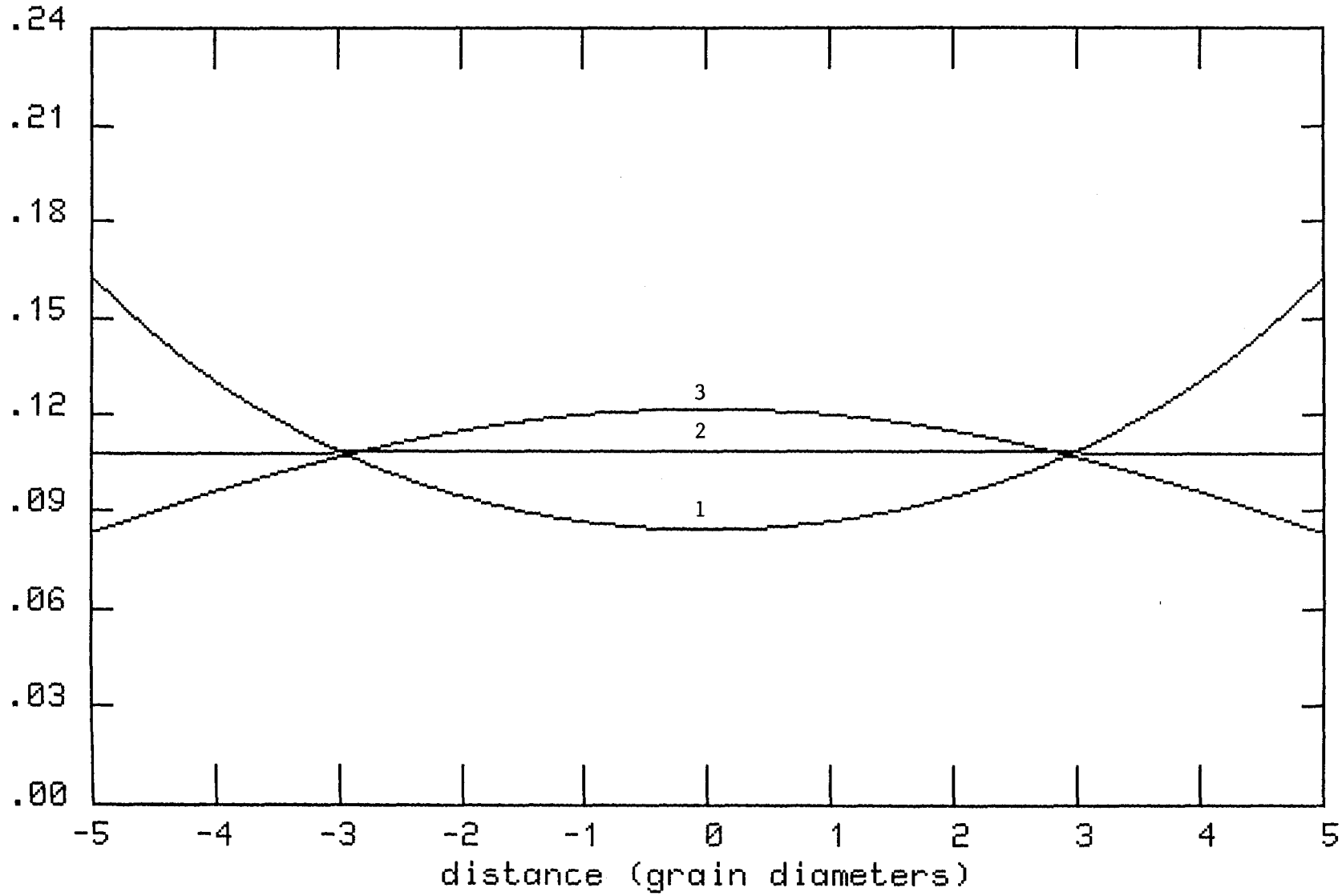



Figure 2.10c: Ratio of particle-particle separation to particle diameter

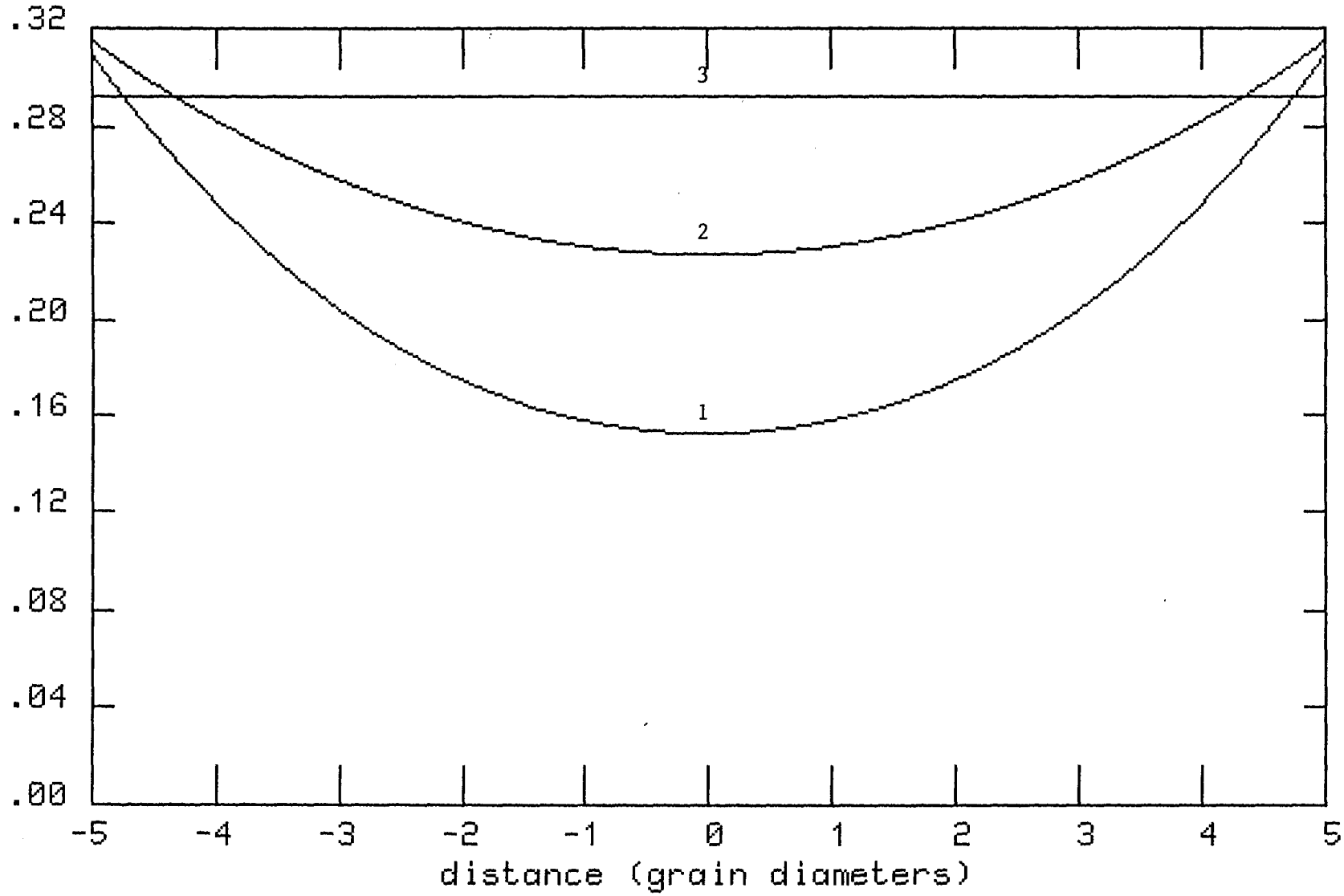



Figure 2.11a: Ratio of particle thermal velocity to wall velocity

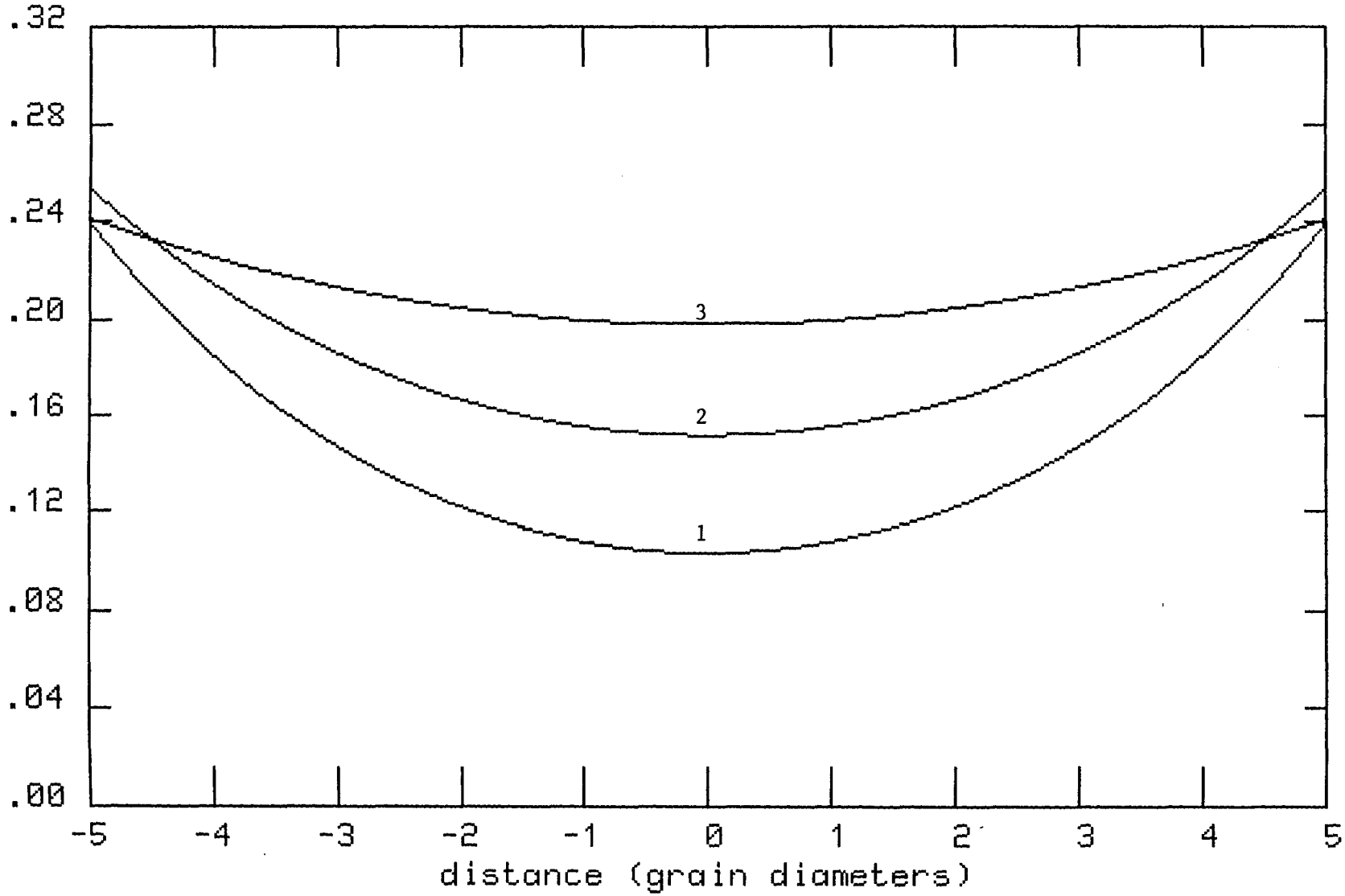



Figure 2.11b: Ratio of particle thermal velocity to wall velocity

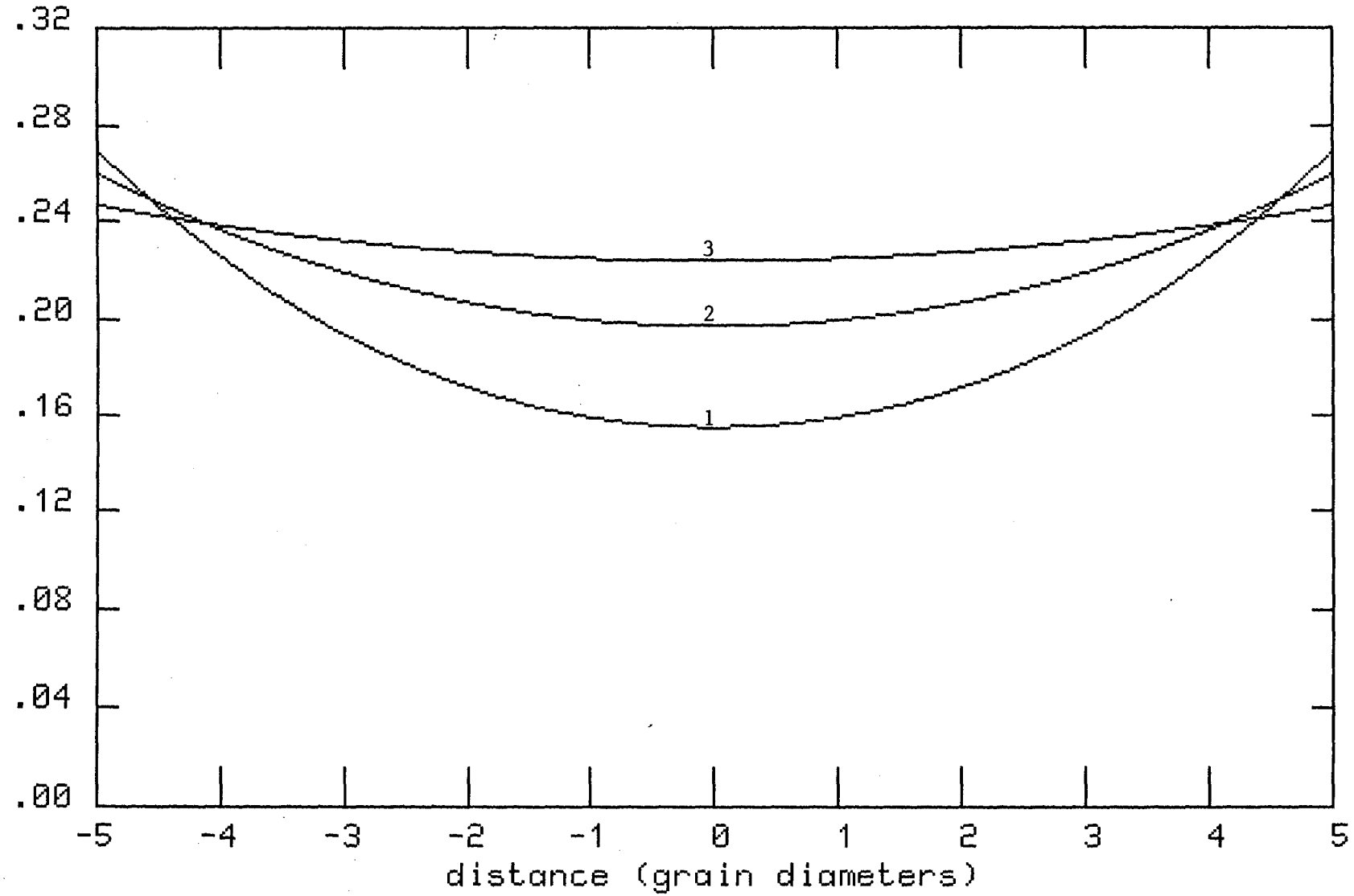



Figure 2.11c: Ratio of particle thermal velocity to wall velocity

# Chapter 3. The Lattice Grain Dynamics Paradigm

In the previous two chapters, the theoretical treatment presented used a traditional approach: the granular system was viewed as a continuum and was modelled using differential equations. However, as was noted earlier, the theories given thus far only apply to highly agitated systems in which the particles interact through binary collisions. Other continuum theories, which are part of the science of soil mechanics, cover the case of static groupings of granular materials. Although a preliminary attempt has been made to obtain a differential equation formalism describing granular systems which covers the complete range of effects from static piles of grains to highly dynamic flows (Johnson and Jackson (1987)), it only patches together in an *ad hoc* manner the existing theories related to the static and the highly dynamic cases. A fully developed differential equation formalism, grounded in the known micromechanics of grain-grain interactions, remains to be formulated. In addition, differential equations assume that the granular system can be treated as a continuum; many problems of interest involve effects which depend upon the macroscopic size of the individual grains (*e.g.*, in the avalanche of a granular material down an inclined slope, the upper surface may consist of a layer of saltating grains, and the shearing layer between the bed and the bulk of the flow may only be a few grain diameters thick) and thus bring this assumption into question. Therefore, in this chapter, I will propose a new formalism for the theoretical description of granular systems based on the techniques of particle dynamics and cellular automata and called lattice grain dynamics.

The purpose of lattice grain dynamics is to predict the behavior of large numbers of grains (10,000 to 1,000,000) on scales much larger than a grain diameter. In this respect it goes beyond particle dynamics calculations which are limited to no more than $\sim 10,000$ grains by currently available computing resources (Walton (1984), Werner (1987)). Unlike particle dynamics, the positions and velocities of individual grains are not calculated precisely but only approximately, based on the premise that large scale effects are relatively insensitive to the exact trajectories of

individual grains. Recent work in applying cellular automata to the modelling of fluid mechanics problems (Frisch *et al.* (1986); Margolis *et al.* (1986)) ("lattice gas dynamics") supports this premise and was the starting point for the development of lattice grain dynamics.

### The Particle Dynamics Method

The particle dynamics method (Cundall and Strack (1979)) is an adaptation of the molecular dynamics method (Alder and Wainwright (1959)) and is designed to study the time evolution of a group of particles by following the motion of each individual particle. The exact formulation of the method depends upon the model adopted for particle-particle interactions. In one formulation, the interparticle contact times are assumed to be of finite duration, and each particle may be in simultaneous contact with several others. In this case, a table is maintained of all of the particle masses, positions, and velocities, and is evolved forward in time in accordance with the interparticle and gravitational forces using an appropriate numerical integration scheme (Werner (1987)). This technique is most useful in studying problems which may depend upon enduring particle-particle contacts (*e.g.,* sorting of granular materials by size, shape, or surface roughness, or when detailed information on particle packing, distribution of stresses, *etc.* is desired). It is, however, a computationally intensive algorithm, which limits the number of particles in a simulation to about 1000.

In another, simpler formulation, the interparticle contact times are assumed to be of infinitesimal duration, and particles undergo only binary collisions (Campbell (1982)). Here, from the table of particle masses, positions, and velocities, a list is derived which contains the time to collision for each possible pair of particles, ordered from shortest to longest. The positions of all of the particles are updated using a time step equal to the time to the next collision (from the top of the list). A model for the interaction of two colliding particles is chosen (*e.g.,* the hard disk model of Appendix 2) and is used to calculate the results of the next collision. Then the entries on the list relating to the two particles which just collided are

discarded and new entries are generated according to the new velocities of those particles. Finally, the list is again ordered from shortest to longest time to collision, and the process is repeated. This method can only be used in systems where the contacts between particles are of short duration compared to the time between collisions (*e.g.,* flow of grains down a vertical chute) (see Walton (1984)); however, due to its simpler computational requirements, this method can handle simulations of up to 10,000 particles. The desirability of using this simpler interaction model, while still being able to handle (in some fashion) enduring contacts, was one of the considerations in the creation of lattice grain dynamics.

In both of these formulations, the time required to find all possible interparticle contacts would normally be proportional to $N^2$, the square of the number of particles, since the position of each particle would have to be compared to the position of every other particle. This can be reduced to a time approximately proportional to $N$ by dividing up the physical space of the problem into many small cells, each containing only a few particles, and looking at potential contacts between particles within the same cell and within adjacent cells. Carrying this idea to the limit of one particle per cell leads to the concept of cellular automata.

### Cellular Automata and The Standard Lattice Gas Model

The concept of cellular automata as a means of modelling physical systems was first proposed by von Neumann in 1948 (Vichniac (1984)). In this theory, the space of a physical problem would be divided up into many small, identical cells each of which would be in one of a finite number of states. The state of a cell would evolve according to a rule which is both local (involves only the cell itself and nearby cells) and universal (all cells are updated simultaneously using the same rule). Originally, cellular automata were applied to the evolution of biological systems; only recently have they been proposed as a formalism for physical theories.

The standard lattice gas model involves the use of large numbers of particles which are constrained to lie on the vertices of a two-dimensional triangular lattice (three-dimensional problems are discussed below). In order to embody both the

kinematic and dynamic aspects of fluid mechanics, the particles are given properties similar to those of a molecule in an ideal gas: each particle is propagated through the lattice according to its velocity and may interact with other particles through elastic collisions. The rules governing the system may be summarized as follows:

1. Each particle has a unit velocity vector which points in one of six directions corresponding to the lattice directions (figure 3.1).
2. A single vertex may have at most only one particle for each velocity direction; thus the state of a vertex (or automata) may be stored using only 6 bits.
3. Each equal time step consists of two stages: an updating of the positions of the particles and an evaluation of their collisions.
4. Each particle advances exactly one lattice spacing in each time step in a direction determined by its velocity vector.
5. Two or more particles on the same vertex may collide according to a simple set of rules; a typical set is given in figure 3.2. Since the state of a vertex is described by 6 bits, the number of possible states is only $2^6$ or 64; and the number of possible pairs of initial and final states is $2^{12}$ or 4096. If we assign a probability to each pair of initial and final states (*i.e.,* collisions), it is feasible to simply store these probabilities in a lookup table, rather than performing a calculation for each collision, thus achieving a computationally more efficient model. In addition, the model now requires no arithmetic calculations at all, allowing the construction of very simple, specialized computers for the solution of lattice gas problems (Margolis *et al.* (1986)).

stress in a given direction, a particle must have both a component of velocity in that direction and a component of velocity perpendicular to that direction. Thus, a shear stress cannot be transmitted parallel to the rows or columns of a square lattice, but can be transmitted, for example, at a 45 degree angle to the lattice. This would have the unphysical effect of introducing preferred directions into the problem, based on the underlying lattice. It can be seen that in a triangular lattice, a shear stress can be transmitted in any direction.

2. The single value of velocity magnitude and the limited number of velocity directions are designed to keep the position updating and collision rules as simple as possible (so as to speed simulation of the system) as well as to keep the memory requirements per vertex as small as possible (only six bits). Also, the single velocity magnitude means that the kinetic energy of the particles is tied to the number of particles, and thus energy conservation is ensured by simple particle number conservation. However, one shortcoming of the single velocity magnitude is that the system has one constant temperature throughout time and space. Thus, this formulation of fluid mechanics can only be used on incompressible systems at low (<.5) Mach numbers and with no temperature gradients. ("Temperature" in a granular system (as defined in Chapters 1 and 2 and in Haff (1983)) and its gradients are an important aspect of the modelling of those systems.)
3. The choice of rules for the evaluation of collisions is guided by the need to conserve the two components of momentum. If only two-particle interactions are included in the set of collision rules, then we would have conservation of momentum along all three lattice directions. The three-particle collision rule (figure 3.2b) is designed to remove this overconstraint while retaining the conservation of orthogonal components of momentum.
4. In a gas at standard temperature and pressure, the mean free path of a molecule is much greater than the effective diameter of a molecule in a collision (*e.g.*, for air at standard temperature and pressure the effective diameter is $\sim 2 \cdot 10^{-8}$ cm and the mean free path is $\sim 2 \cdot 10^{-5}$ cm). This means that the direction of

the impact parameter (the unit vector pointing from the center of particle 1 to the center of particle 2 at the moment of contact) (figure 3.3) will be essentially independent of the starting positions of the molecular trajectories. Thus, the choice of collision rules may be made without regard to the impact parameter, as long as they do not require an unreasonable distribution of impact parameter angles (*e.g.*, in the collision rules represented by figure 3.2a, the impact parameter angles were fixed at +60 degrees to the relative velocity vector for half the collisions, and at −60 degrees for the other half of the collisions) (see Goldstein *et al.* (1989)).

The memory requirements for a typical two-dimensional lattice gas simulation involving a few million lattice points will be a few megabytes (one byte per lattice point); this amount of memory has now become economically available on workstation-type computers. Due to the simplicity and efficiency of these rules, the updating of a lattice point requires only a few tens of machine cycles; thus a typical workstation (operating at a few million instructions per second) can update approximately 100,000 lattice points per second. Combining these two facts, we see that problems involving large numbers of lattice points (1,000,000) may be evolved for significant lengths of time (thousands of time steps) on relatively modest size computers.

The formulation of a lattice gas paradigm for three-dimensional problems is not trivial. In order to avoid the problem of inadequate symmetries of the pressure tensor referred to above, a simple cubic lattice cannot be used. Also, there are no three-dimensional lattices which have properties similar to the triangular lattice in two dimensions (*i.e.*, all vertices are equidistant from each other and all square symmetries are broken). Two possible solutions which have been proposed by d'Humieres *et al.* (1986) involve the use of a four-dimensional lattice projected onto three dimensions or the use of a three-dimensional face-centered cubic lattice with multiple particle velocities. The number of bits needed to describe the state of a vertex (automata) are 24 and 19 respectively; the large number of collision

possibilities obviates the use of a lookup table for obtaining the results of a collision. Thus, very little progress has been made towards extending lattice gases to three-dimensional problems.

### Derivation of the Rules for Lattice Grain Dynamics

Inasmuch as the use of cellular automata for the formulation of physical theories is a recent development, the methods by which the rules governing the automata are obtained are not well established. Although some goals (*e.g.,* conservation of momentum) may be obvious, other rule-determining factors will only become apparent after trial and error work with the automata. Thus, the motivations behind the rules as well as the rules themselves will be treated in some detail in this section.

The formulation of lattice grain dynamics is based upon considerations similar to those for lattice gases, modified by two unique characteristics of granular systems: the inelasticity of grain-grain collisions and the usually short length of the mean free path relative to a particle diameter. The short length of the mean free path means that the transmission of forces and momenta through the system will be dominated by collisions and enduring contacts, as opposed to a standard density gas in which kinetic transport dominates. In order to keep the particle-particle interaction rules as simple as possible, all interparticle contacts, whether enduring contacts or true collisions, will be modelled as collisions. Those collisions which model enduring contacts in each time step will transmit an impulse equal to the force of the enduring contact times the time step. The large size of a particle means that the impact parameter in a collision will be determined by the relative locations of the particles involved and can no longer be set arbitrarily. This, as well as the inelasticity of collisions, forces us to allow for a range of velocity magnitudes. In addition, it is no longer possible to permit more than one particle on a lattice point; instead we must allow for collisions between particles on adjacent lattice points. Although these considerations will mean that the lattice grain dynamics paradigm will be slower to compute than the standard lattice gas paradigm, it will nonetheless be more efficient than particle dynamics calculations.

*The time step*

The determination of the size of the time step must be made more carefully than in the standard lattice gas theory due to the many possible particle velocities. If the time step is long enough that some particles travel several lattice spacings in one time step, there arises the problem of finding the intersections of particle trajectories. This time-consuming procedure would eliminate much of the efficiency of the lattice approach. On the other hand, a very short time step would imply that most particles would not move even a single lattice spacing in that time step. The solution adopted here is two-fold in nature. The time step is chosen such that the fastest particle will move one lattice spacing in that time step. In order to insure that a slower velocity particle will eventually move a lattice spacing, its "position offsets" are accumulated until they equal one lattice spacing, at which point the particle is promoted to the next site. Thus, for the case with no gravity, the time step is given by the lattice spacing divided by the speed of the fastest particle. For the case with gravity, the acceleration of the fastest particle during the time step must be taken into account. This leads to a fourth order polynomial for the value of the time step (see Appendix 1), the solutions to which are shown graphically in figure 3.4. The solution of interest (the smallest of the positive solutions) is obtained by successive numerical approximations.

*Position offsets and particle motion*

The two components of the position offset of each particle are updated at the beginning of the time step according to the components of its velocity and gravitational acceleration:

$$\Delta q_i = v_i \Delta t + \frac{1}{2} g_i \Delta t^2,$$

where:

$i = 1, 2,$

$\Delta q_i =$ i*th* component of increment in position offset,

$v_i =$ i*th* component of particle velocity,

$g_i =$ i*th* component of gravitational acceleration,

$\Delta t =$ current time step.

If the magnitude of the position offset exceeds a given value, the particle is eligible to be moved to the lattice point nearest to the offset (figure 3.5); and if the destination lattice point is empty, the particle is moved and its offset is appropriately decremented.

The question arises as to what should be done with the position offset of a particle which undergoes a collision. If the offset is left unchanged after a collision, the situation may occur in which the particle's velocity has been changed, but the particle continues to move in its original direction due to its accumulated position offset. If both components of the offset are set to zero in a collision, a particle which has a small component of velocity perpendicular to the direction of the collision will never be able to move in that perpendicular direction. This situation can occur in association with a gravitational field, where the collision is supporting the particle against the field, with the unphysical result that tall stacks of particles will be stable (figure 3.6). Experience has shown that the best solution is to reset the component of position offset along the direction of a collision to zero, while leaving the orthogonal component unchanged.

It is important to consider the order in which the lattice is scanned for this position updating process. A simple scan (*e.g.,* following successive points in a row, figure 3.7) will result in an undesirable coupling between the scan pattern and the particle motions. To see this, consider the one-dimensional example shown in figure 3.8. In 3.8a, the direction of scan coincides with the direction of motion of the line of particles, with only the particle at the right end being moved. In 3.8b, the direction of scan is opposite to the direction of motion of the particles, with the result that the entire line moves one space. In order for the particle motion to be independent of the scanning process, a more complex scan pattern needs to be adopted. One possibility would be to scan every odd particle first and then go back and do the even particles. This scheme, however, still runs into a problem in the case of two particles contending for the same empty lattice point (figure 3.8c). Which particle will be able to move is again determined by the direction of scan. Thus, the scheme adopted for the lattice grain dynamics paradigm is to do every

third lattice point in every third row, duplicating this pattern nine times in order to cover all lattice sites (see figure 3.9). In addition, the entire position update process may be repeated several times per time step (the number of updates per time step is an input parameter in the specification of a simulation, and is typically set to three). This approach allows the position updating of different parts of the lattice to be done in parallel, an important consideration when implementing this procedure on an array of parallel computers.

*Particle collisions*

Collisions between particles are calculated assuming that they are smooth, hard disks and that their coefficient of restitution is velocity dependent. Since a particle may be surrounded by as many as six other particles, and may in principle collide with all six in one time step, some rule is needed for determining the results of such multiple particle collisions. In order to keep the rules simple, it was decided that a multiple particle collision would be resolved as a series of binary collisions done in a specified order. This order must be chosen carefully so as not to create any bias in any direction associated with the lattice. If, for example, the order of binary collisions with adjacent particles were the same for each lattice point, then the collision between two lines of particles (figure 3.10) would not produce a random scattering of particles as expected, but would instead send each line in a preferred direction. Also, a bias will be introduced if the order of collisions at a given lattice point is the same for several successive time steps.

The order in which the lattice points are scanned (for evaluating all the collisions in one time step) must be chosen carefully as well. As was the case for the particle movements, a simple scan of successive points along a row will give results which depend on whether the momentum transfer is parallel to or antiparallel to the direction of scan. Here, the problem may be solved by scanning every other lattice point on every other row, and repeating this pattern four times. If three non-parallel collisions are evaluated at each lattice point with this scanning pattern, then all possible collisions will be evaluated once (figure 3.11). Thus, the following order has been adopted for evaluating possible collisions on odd time steps: 3b, 3c, 3f, 2f,

2c, 2b, 4b, 4c, 4f, 1f, 1c, 1b; and for even time steps: 1b, 1c, 1f, 4f, 4c, 4b, 2b, 2c, 2f, 3f, 3c, 3b. Note that both the order of the scanning templates and the order of collisions at a given lattice point are reversed for successive time steps. The results of this scanning order are given in the following table for collision order at each lattice point as a function of time step and position in the lattice.

| time step | lattice position | order of collisions | | | | | |
|---|---|---|---|---|---|---|---|
| odd | 1 | e | d | a | f | c | b |
| | 2 | a | f | c | b | e | d |
| | 3 | b | c | f | a | d | e |
| | 4 | d | e | b | c | f | a |
| even | 1 | b | c | f | a | d | e |
| | 2 | d | e | b | c | f | a |
| | 3 | e | d | a | f | c | b |
| | 4 | a | f | c | b | e | d |

The derivation of the velocities of two particles after a collision as a function of their initial velocities and the impact parameter is given in Appendix 2. The velocity dependent coefficient of restitution is defined such that the slope of the relative rebound velocity versus relative incident velocity curve has one given value ($e_1$) up to a specified relative incident velocity ($c_b$), and has a separately defined value ($e_2$) above that point. The relative rebound velocity as a function of relative incident velocity is plotted in figure 3.12 and is given by:

$$c_r = \begin{cases} e_1 c_b + e_2(c_i - c_b), & \text{if } c_i > c_b; \\ e_1 c_i, & \text{otherwise.} \end{cases}$$

where:

$c_b$ = velocity below which coefficient of restitution is $e_1$,

$c_i$ = magnitude of relative incident velocity,

$c_r$ = magnitude of relative rebound velocity,

$e_1$ = initial coefficient of restitution,

$e_2$ = incremental coefficient of restitution.

The transmission of "static" contact forces within a mass of grains (as in grains at rest in a gravitational field) is handled naturally within the above framework. At the beginning of each time step, each particle's velocity is incremented due to the acceleration of gravity:

$$\Delta v_i = g_i \Delta t$$

Thus, in the lattice grain picture, even though a particle in a static mass of grains may nominally be at rest, its velocity will be nonzero; and it will transmit the appropriate force (in the form of an impulse) to the particles under it by means of collisions. When these impulses are averaged over several time steps, the proper weight of the particle will emerge.

*Wall particles*

Many problems of interest in granular physics involve some type of container, barrier, or wall which restricts the movement of the grains, as in flow down an incline or flow around an obstacle. In order to incorporate a wall within these rules, a second type of particle is introduced: the wall particle. This particle is similar to the movable particles, and interacts with them through binary collisions, with a separately defined inelasticity, but is regarded as having infinite mass. This infinite mass of a wall particle allows for problems involving walls or obstacles which are not moved by the flowing particles and which can absorb whatever momentum the flow may transfer to them. Because a wall consists of many particles of the same size as a movable particle, even a nominally flat wall (*i.e.*, a straight line of wall particles) (figure 3.13a) exhibits frictional effects when in contact with a shearing type of flow. A wall may be further roughened by arranging the wall particles in some pattern other than a straight line (figure 3.13b). To allow for the introduction of shearing motion from a wall (as in a Couette flow problem), the wall particles are given a common constant velocity, which is used in the usual fashion for calculating the results of collisions. However, the position of the wall particles in the lattice remains fixed throughout the simulation. The sum of all the collision impulses received by a wall particle during a time step can be resolved into an effective pressure and shear stress exerted on that wall particle.

*Physical interpretation of the particles*

A final question concerns the interpretation of what a cellular automata particle represents in this model. In the standard lattice gas paradigm, the individual particles do not represent any real world object. Although they possess some qualities similar to those of the molecules of a gas (*e.g.*, mean free path much greater than a particle diameter and perfectly elastic collisions), they are physically quite distinct from molecules (*e.g.*, they have one fixed velocity magnitude and a mean free path and mass much greater than that of a molecule). On the other hand, a lattice particle does not represent a small, fixed volume of the gas, since it can pass freely by particles on adjacent lattice points without interacting with them. The particles of the standard lattice gas are thus fictitious. The motion of individual particles is not physically replicated in the real world; they are useful, however, because the average behavior of a small group of lattice particles models the average behavior of a large number of molecules (or a small volume of gas). In the case of lattice grain dynamics, the proposed interpretation of a particle is simpler and more physical. Even though a single particle does not accurately predict the trajectory of a single grain, we nonetheless regard each particle as representing one grain when we are extracting information from the simulation regarding the behavior of groups of grains. Thus, the size of one particle, as well as the spacing between lattice points, is taken to be one grain diameter. This means that the exclusion principle given above (no more than one particle per lattice point) corresponds to the exclusion of all other grains from the volume of space occupied by a real grain, and that the properties of two particles in collision correspond to the properties of two real grains in collision. This interpretation also embodies the fact that a static group of grains can only be compressed so far; once the grains develop a network of enduring contacts, it becomes very difficult to compress them further.

## Implementation of the Algorithm on a Single Processor Computer

*Roundoff error and integer arithmetic*

One of the questions which arises in the implementation of a classical physics algorithm on a digital computer is the issue of roundoff error. In classical physics

treatments of particle motion (as in particle dynamics), the particle positions, velocities, and accelerations, as well as time, are regarded as analog quantities which are infinitely divisible. The digital computer, on the other hand, can only work with quantities which have a finite number of values (*i.e.*, the computer's numerical representation of these quantities in its memory can have only a finite number of significant digits). This requires the computer to round off the values of these quantities to the nearest least-significant digit, thus creating a roundoff error in the model computations. If this rounding-off process is not done carefully, it may introduce a biased error into one or more of these quantities, leading, for example, to a gradual gain or loss of energy in a system in which energy should be exactly conserved. One of the advantages of the standard lattice gas is the fact that all of the quantities used in the model are restricted to one or a few discrete values and thus may be represented exactly in the computer's memory. Consequently, momentum and energy conservation in numerical evaluations of the standard lattice gas model are always exact.

In the lattice grain dynamics paradigm, the particle positions and velocities are calculated only approximately; thus it is unnecessary to represent these quantities with floating-point numbers, which are capable of storing fractional values. Instead, it is feasible to rescale all quantities of interest (position offsets, velocities, and time) so as to be able to represent them with integer numbers. This gains two advantages in the numerical evaluation of this model: in the transfer of momentum between two particles in a collision, the initial velocities as well as the changes in velocities are all integers, thus maintaining an exact conservation of momentum; and the processing of integer arithmetic in a computer is generally faster than the processing of floating-point arithmetic (*e.g.*, integer addition on a Tektronix 6130 workstation is over twice as fast as floating point addition and over three and a half times as fast as double precision addition). Therefore, the two components of position offset and velocity of a particle on a lattice point are stored as four integers. Although the energy of the system can still vary due to roundoff error, this is not a significant problem, since energy is not conserved in grain-grain collisions. The

presence of even a biased roundoff error in the calculations means only that the rate of energy loss will be slightly higher or lower than was expected based on the coefficient of restitution and the rate of collisions.

Finally, it should be noted that, in a collision, the vector change in velocity will lie along the line connecting the two particle centers at the moment of contact. The resolution of this vector into $x$- and $y$-components will involve the values of $\cos 60°$ and $\sin 60°$ if this collision is along the a, b, e, or f directions. Rather than have the computer repeatedly do the time-consuming evaluation of these trigonometric functions, the values of $\cos 60°$ and $\sin 60°$ are built into separate portions of code for each of the possible collision directions. In order to remain in integer arithmetic, these fractional values are obtained by multiplying by the ratios 780/1560 and 1351/1560, respectively.

*Memory requirements*

When implementing this algorithm on a computer, what is stored in the computer's memory is information concerning each point in the lattice, regardless of whether or not there is a particle at that lattice point. This allows for very efficient checking of the space around each particle for the presence of other particles (*i.e.*, information concerning the six adjacent points in a triangular lattice will be found at certain known locations in memory). This is in contrast to the particle dynamics method in which a list of particles is maintained, and the position of a particle must be compared with the positions of all the other particles in the list in order to find its neighbors (Werner (1987)). Thus, the storage requirements and, to some degree, the computational load are proportional to the number of lattice points under consideration, rather than the number of particles. The need to keep information on empty lattice points in memory does not entail as great a penalty as might be thought; many lattice grain dynamics problems involve a high density of particles, typically one for every one to four lattice points, and the memory cost per lattice point is not large. The storage cost per lattice point amounts to five integer variables: two components of position offset, two components of velocity, and one status variable, which notes whether the lattice point is empty or contains

a movable particle or a wall particle (it also allows for the designation of a small number of movable particles as marked particles whose motion can be stored and later plotted). If each integer is stored using four bytes of memory, then each lattice point requires 20 bytes of memory. So, for example, the storage of a four million lattice point simulation would require 80 megabytes of memory, an amount which is now available on single processor workstations and concurrent processor computers.

*Problem specification*

The standard configuration for a simulation consists of a lattice, with a specified number of rows and columns (from four to several thousand rows and from six to several thousand columns), bounded at the top and bottom by two rows of wall particles (thus forming the top and bottom walls of the problem space), and with left and right edges connected together to form periodic boundary conditions (*i.e.*, a particle passing out through one edge will reappear at the other edge at the same height). The periodic boundary conditions are achieved by simply regarding the rightmost column of the lattice as being adjacent to the leftmost column. Thus the boundaries of the lattice are handled naturally within the normal position updating and collision rules, with very little additional programming. The only restriction is that the number of columns must be a multiple of six in order to accommodate the periodicity of the position scanning pattern (two) and the periodicity of the collision scanning pattern (three) within the periodic boundary conditions in the $x$-direction. (Note: since the gravitational acceleration can point in an arbitrary direction, the top and bottom walls can become side walls for chute flow. Also, the periodic boundary conditions can be broken by the placement of an additional wall, if so desired. Simulations involving free surfaces can be run by leaving a large number of empty rows above the free surface in the specification of the initial conditions, so that particles will not hit the upper wall.)

Two input files are required: one setting physical properties data and the other giving the initial particle positions and velocities. The first file specifies the properties of the movable and wall particles (inelasticities, *etc.*), the components of the gravitational acceleration, the shearing velocities of the top and bottom walls (if

any), the number of time steps, the number of position update scans per time step, and the number of rows and columns in the lattice. A second input file contains the initial positions and velocities of all the movable particles, any marked movable particles, and any additional wall particles which may be desired. Rectangular areas of particles with one of two densities (one particle in every lattice point or one particle in every three lattice points) may be specified by giving the coordinates of the lower left corner and the upper right corner. A straight line of particles may be specified by giving the coordinates of the two endpoints of the line. (As was noted before, a straight line of wall particles will act as a rough wall, not as a smooth wall.) The units used in the program are not related to any physical units. Instead, the velocities, positions, and time step increments are scaled so as to fall in the range of 1 to 32,767 in order to allow the use of integer arithmetic and storage. This method of specifying the problem has proven to be fast and efficient, and allows for a wide range of problems to be solved (as will be demonstrated in the section on simulations). An initial condition file typically consists of 10 to 100 lines and generally can be set up in less than an hour. Examples of initial particle positions are shown in figures 3.16, 3.18, 3.20, 3.24a, 3.28, and 3.33.

### Implementation of the Algorithm on a Concurrent Processor Computer

Inasmuch as a particle may move no more than one lattice space in a time step and may collide with only the particles on its six adjacent lattice points, the updating of the lattice need not be done in a serial fashion; instead, different portions of the lattice may be processed in parallel. The lattice may be divided up into roughly equal area sections, with the calculations in each section handled by its own processor. The only interaction between sections will be along their common boundaries, thus each processor will only need to exchange information with its eight immediate neighbors (figure 3.14). The instructions being executed in each processor will be very similar; however, the processors at the top and bottom edges of the lattice will have to handle their boundaries somewhat differently than the rest. This collection of processors is called a multiple instruction, multiple data (MIMD)

concurrent processor computer (CPC), since each processor has its own program and its own separate data memory. The processors and their interconnections are arranged in the form of a hypercube, with the processors forming the nodes and the interconnections forming the edges. With this hypercube topology, the number of communications channels per processor is only a logarithmic function of the number of processors, and it is feasible to connect together as many as 1024 processors in a 10-dimensional hypercube configuration. One additional computer, the host computer, is required for handling communications between the hypercube array and the magnetic data storage disks and terminals, for the purpose of loading the programs and input data into the nodes of the array and storing and displaying the results. The simulations reported in the next section were done on a CPC built by NCUBE Corporation which contained 512 processors. In this machine, each processor node consists of one very large scale integrated circuit, containing all of the required logic for computation and communications, and six memory chips, giving 512 kilobytes of memory.

For the purpose of dividing up the problem among the many processors of the CPC, the hypercube architecture is unfolded into a two-dimensional array, and each processor is given a fixed region of the lattice (figure 3.15). In addition, each processor stores information relating to one extra row at the top and bottom edges and one extra column at the left and right edges of the portion of the lattice for which it is responsible. This information is updated several times during each time step of the simulation by means of communications between adjacent processors (figure 3.14). These communications consist of a copy of all the lattice point data for one edge row or one edge column of one processor to be stored in the extra row or column of the adjacent processor. The only communications required among all the processors (global communications) occur in the determination of the size of the global time step and the accumulation of some global statistics (average velocities, total energy, and total wall pressures).

The program itself was written in the C programming language under the Cubix/CrOS III operating system. With Cubix, only a program for the nodes of

the hypercube CPC needs to be written; no separate program for the host computer is required. The activities normally required of the host computer (communications to and from disks and terminals) are built into the input/output routines called by the node programs. This resulted in a significant reduction in the amount of time needed to write and debug the program for the CPC. The use of the C programming language in combination with the communication routines provided in CrOS III allowed for the creation of particularly simple and elegant code for communications between the nodes of the CPC. With C, the data associated with the lattice was stored as an array of data structures, where each structure consisted of the five integer variables associated with one point on the lattice. Thus, the information concerning a lattice point could be sent as a single group, rather than as five separate integers. Only a single subroutine call was needed to send the information regarding an edge row or an edge column between adjacent nodes. To send the data for the edge row of one node to another, a call was made to the `cshift` function in each node, specifying the starting address of the data, the number of bytes to be sent, and the number of the destination node (in the source node) or the number of the source node (in the destination node). The transmission of data concerning edge columns used the `vshift` function in a similar manner.

## Simulations

### *Testing the program*

A series of ten simple problems involving from one to twenty particles were used to test the program to ensure that the calculations were being performed correctly and that no particles or momenta were gained or lost when transferred from one CPC node to another. In addition, the results of these simulations were compared with the results obtained from a version of the program written in Fortran and running on a single processor workstation. The only differences seen were due to known minor differences in the programs; no real problems were found.

### *Testing the lattice grain dynamics paradigm*

The next test of the lattice grain dynamics paradigm was a simulation of a two-dimensional gas of elastic particles in a box. This test was chosen because an

analytic expression for the equilibrium velocity distribution of a two-dimensional gas is known; thus this problem tests both the ability of the lattice grain dynamics paradigm to accurately model the physics of a real-world problem as well as the correctness of the computer code. A total of 4096 particles were placed in a box of 33,024 open lattice points, arranged in 128 rows of 258 lattice points per row, with two rows (at the top and bottom) of wall particles (figure 3.16a). Their initial velocities were all of the same magnitude, with half the particles moving in the positive $x$-direction and half moving in the negative $x$-direction (figure 3.16b). The simulation was run until equilibrium was established and a plot of the result is shown in figure 3.17a. It is expected that the particle velocities in equilibrium will have a Maxwellian distribution appropriate to two dimensions. Each of the two components of velocity will have a distribution of:

$$f_x = f_y = \frac{n}{\sqrt{\pi \overline{v^2}}} exp\left(\frac{-v^2}{\overline{v^2}}\right).$$

While the peculiar speed distribution will be:

$$f_s = \frac{2nv}{\overline{v^2}} exp\left(\frac{-v^2}{\overline{v^2}}\right).$$

where:

$v$ = velocity magnitude,

$\overline{v^2}$ = mean of the square of the velocity,

$f_x(v)dv$ = number of particles with $v_x$ within $dv$ around $v$,

$f_y(v)dv$ = number of particles with $v_y$ within $dv$ around $v$,

$f_s(v)dv$ = number of particles with speed within $dv$ around $v$,

$n$ = number of particles.

The measured and theoretical distributions for the two components of velocity as well as the magnitude of velocity are shown in figure 3.17b and are in good agreement.

*An explosion*

This simulation was suggested by a test of a simple lattice gas algorithm by Margolis *et al.* (1986). In their two-dimensional lattice gas model, all particles travel with constant speed in one of only four allowed directions; particles which collide head on scatter at right angles. Their test consisted of a 256 by 256 lattice filled to a density of one-half, except for a central 32 by 32 region filled to a density of one, with no gravity. As the system evolved forward in time, this square block of high density became a circular ring of high density which expanded outward uniformly in all directions, despite the square symmetry of the underlying lattice and rules.

In the simulation run here, a 288 by 288 lattice was filled with elastic, initially motionless particles to a density of one-third (figure 3.18a); the gravitational acceleration was set to zero. A 33 by 35 rectangular set of particles was placed in the center, and the particles were given initial velocities along the $x$- or $y$-directions (an "explosion") (figure 3.18b). The simulation was run for 128 time steps to produce the result shown in figure 3.19a; the front of the shock has become a circle expanding uniformly in all directions. The lattice was divided up into small 8 by 8 regions, with the particle velocities in each region averaged together and displayed as a single arrow (figure 3.19b). The velocities near the edge of the shock are seen to be oriented radially outward. Thus, the underlying triangular lattice and the limited number of orientations for a collision have not introduced any asymmetries into the simulation.

*Couette flow*

The first test simulation of a moving granular system was performed using a two-dimensional Couette flow geometry. The standard Couette flow configuration consists of a fluid confined between two, flat, parallel plates of infinite extent, without any gravitational accelerations. The plates move in opposite directions with velocities that are equal and that are parallel to their surfaces, which results in the establishment of a velocity gradient and a shear stress in the fluid. For fluids which obey the Navier-Stokes equation, an analytical solution is possible in which the velocity gradient and shear stress are constant across the channel. If, however,

we replace the fluid by a system of inelastic grains, the velocity gradient will no longer necessarily be constant across the channel (see Haff (1983)). Instead, what often occurs is the creation of a plug flow region in the center of the channel with two highly shearing regions near the plates.

The simulation was carried out with 5760 grains, located in a channel 60 lattice points wide by 192 long (figure 3.20). Due to the periodic boundary conditions at the left and right ends, the problem is effectively infinite in length. The top and bottom wall particles were given equal but opposite velocities along the direction of the wall; thus, by symmetry, the average velocity of the center of the flow will be zero. The first simulation is intended to reproduce the standard Couette flow for a fluid; consequently the particle-particle collisions were given a coefficient of restitution of 1.0 (*i.e.,* perfectly elastic collisions) and the particle-wall collisions were given a .75 coefficient of restitution. The inelasticity of the particle-wall collisions is needed to simulate the conduction of heat (which is being generated within the fluid) from the fluid to the walls. The simulation was run until an equilibrium was established in the channel, figure 3.21a. In figure 3.21b are shown plots of the average (along the channel) $x$- and $y$-components of velocity, as well as the second moment of velocity (the square of which is proportional to the temperature), as a function of distance across the channel. The $x$-component of velocity is a linear function of distance across the channel. The second moment of velocity is seen to peak in the center of the channel and to fall off towards the walls, corresponding to the generation of heat within the fluid which is conducted away to the walls (which act as heat sinks).

The question arises as to whether this simulation represents laminar or turbulent fluid flow. To answer this question, we will make use of the fact that in laminar flow the shear stress will be proportional to the shear rate, while in turbulent flow the shear stress will be proportional to the square of the shear rate. The simulation was run again with the wall velocities multiplied by a factor of four and with all other parameters unchanged. Figures 3.22a and 3.22b show the pressure and shear stress on the top and bottom walls for the original problem and for the speeded up problem, respectively. The pressure and shear stress are seen to increase by a factor

of 16 from the first run to the second run; consequently the simulation represents a turbulent fluid flow. This result comes about from the fact that the "thermal" velocities of the particles are determined solely by (and therefore scale with) the driving velocity in the problem (in this case, the wall velocity). In real world laminar flow, the thermal velocities of the molecules of the fluid are largely determined by the temperature of the walls, which supply or absorb heat energy as needed to maintain a fixed temperature in the fluid. In order to implement this type of behavior in the lattice grain dynamics paradigm, it would be necessary to modify the particle-wall collisions so as to give the rebounding particles a velocity distribution appropriate to a given wall temperature. Since the goal of the paradigm was to simulate granular systems rather than fluids, this additional complication was not incorporated.

In order to simulate a granular Couette flow, the second simulation used a constant coefficient of restitution of .75 for both the particle-particle and particle-wall collisions. The equilibrium result is shown in figure 3.23a. The average $x$- and $y$-components of velocity and the second moment of velocity, as functions of distance across the channel, are plotted in figure 3.23b. As can be seen from the plots, the flow consists of a central region of particles compacted into a plug, with each particle having almost no velocity. Near each of the moving walls, a region of much lower density has formed in which most of the shearing motion occurs. Note the increase in value of the second moment of velocity (the granular "thermal velocity") near the walls, indicating that grains in this area are being "heated" by the high rate of shear.

*The hourglass*

A second problem studied the flow of grains through a hopper or an hourglass, with an opening only a few grain diameters wide; the driving force was gravity. This is an example of a granular system which contains a wide range of densities, from groups of grains in static contact with one another to groups of highly agitated grains undergoing true binary collisions. Observations of flow in hoppers (Tuzun and Nedderman (1982)) reveal that the granular material may undergo funnel or

core flow, in which the bulk of the grains remain at rest in the hopper while a narrow core of highly shearing flow forms in the center accompanied by a depression in the top surface. The grains flowing down the core come from both the adjacent static regions at the sides of the core region and from grains cascading into the depression at the top. Once the grains leave the hopper, they fall essentially freely and undergo only a few binary collisions before reaching the floor.

Here, the number of particles used was 8310, with $e_1 = .75$, $e_2 = .9375$, and $c_b = \sqrt{2gd}$ (where $g$ = the magnitude of the gravitational acceleration and $d$ = one lattice spacing); and the lattice was 240 points long by 122 wide (figure 3.24a). Additional walls were added to form the sloped sides of the bin and to close off the bottom of the lattice so as to prevent the periodic boundary conditions from reintroducing the falling particles back into the bin. The sides of the bin were sloped at an angle of 41 degrees to the vertical and the opening was set at 12 lattice points. As can be seen in figure 3.24b, the simulation does exhibit the rapidly shearing core region, the static side regions, and the depression at the top characteristic of experimentally observed flows (Tuzun and Nedderman (1982)). However, the slope of the free surface of the grains (*i.e.*, the angle of repose) is greater than what is observed experimentally. This is believed to be due to the two-dimensional nature of the simulation. For a particle in a two-dimensional stack of particles to "escape" from the surface of a steep slope, it must first overcome a substantial gravitational potential barrier in order to get past the particle beneath it (figure 3.25). A particle in a three-dimensional stack can use the third dimension to go around the particle beneath it rather than over that particle. This will lead to a larger angle of repose for two-dimensional systems of grains.

One significant difference between the flow of granular materials from a hopper and the flow of a fluid from the same geometry lies in the rate of flow as a function of time. The flow rate of a fluid will decrease with increasing time (due to the reduction in pressure as the hopper is emptied out); the flow rate of a granular material is relatively independent of the amount of material remaining in the hopper and will be fairly constant with time (this is what made the hourglass a useful timekeeping

instrument). This is shown in figure 3.26 where the total number of grains which have passed out of the hopper is plotted as a function of time. After a gradual startup period, the flow rate (*i.e.*, the slope of the curve) is seen to be nearly constant with time until most of the grains have run out.

*Flows around obstacles*

Another class of problems studied involve the flow of grains around obstacles of different shapes. These flows were observed experimentally by Nedderman, Davies, and Horton (Nedderman *et al.* (1980)) using mustard seeds of .228 cm diameter confined between two glass plates spaced 2.3 cm apart, giving a nearly two-dimensional system. The rate of flow of particles through the apparatus was controlled by placing a flow restrictor at the bottom (figure 3.27); however, the width of this opening was not given. Obstacles of four different shapes were placed in the path of the grains flowing downward under the force of gravity: a circle, a square, a triangle oriented point up, and a triangle oriented point down. The sides of the square and the triangles and the diameter of the circle were all 10 cm; these were centered between sidewalls which were 20 cm apart. Black kale seeds were used as marker particles so as to allow the tracing of flow lines.

The simulation contained 16,384 particles in a lattice of 288 points by 130 points. Inasmuch as the collisional characteristics of the experimental particles were not given, the values of $e_1$ and $e_2$ were determined by varying them so as to obtain the best match between the simulations and the experiments, and were set to $e_1 = .75$ and $e_2 = .9375$, and $c_b = \sqrt{2gd}$, as in the hourglass problem. The flow restrictor at the bottom consisted of a wall across the channel with a 18 lattice point wide opening in the center. Below the flow restrictor a box was placed to catch the used particles. An example of the initial placement of the grains is shown in figure 3.28.

The first of these four simulations involved a circular obstacle, whose diameter was one-half the width of the channel, and which consisted of wall particles placed on lattice points in as close an approximation to a circle as possible. The final results of this simulation are shown in figures 3.29a and 3.29b. A void has formed

under the circular obstacle, as was seen in the experiment (figure 3.29c); and a stagnant region has formed above the obstacle, as can be seen from the motion (or lack thereof) of the marker particles. The void below the circular obstacle is due to the fact that a granular system will not flow horizontally unless its surface is at an angle with the horizontal which is greater than the angle of repose. Thus, the flow follows the underside of the circle until it reaches the angle of repose; after that, it forms its own free surface. Similarly, the top sides of the stagnant region above the circular obstacle are at the angle of repose with respect to the horizontal.

The second obstacle was a square whose side was one-half the width of the channel. The final results of this simulation are shown in figures 3.30a and 3.30b. Once again, the formation of void and stagnant regions corresponds to the experimental observations (figure 3.30c).

The third obstacle was an equilateral triangle whose sides were one-half the width of the channel and which was oriented point up. The final results of this simulation are shown in figures 3.31a and 3.31b. In this case, no stagnation region was seen above the point of the triangle in either the experiment (figure 3.31c) or the simulation (figure 3.31b). This is due to the fact that the sides of the triangle are at an angle of 60 degrees to the horizontal, which is greater than the angle of repose. Thus, the grains are unable to accumulate in the region above the triangle.

The fourth obstacle was an equilateral triangle whose sides were one-half the width of the channel and which was oriented point down. The final results of this simulation are shown in figures 3.32a and 3.32b. Here, the void region below the triangle is larger than was seen in the experiment (figure 3.32c); this is due to the greater angle of repose problem discussed in the hourglass simulation.

*Flow down an inclined slope*

The next problem simulates the flow of grains down an inclined slope and running out onto a flat surface. The initial placement of the grains is shown in figure 3.33, where the gravitational acceleration is perpendicular to the diagonal line of wall particles. Thus the bottom line of wall particles is the sloped region (at an angle of 23.4 degrees to the horizontal) and the diagonal line of wall particles

is the horizontal runout surface. Note that advantage has been taken of the area under the runout surface and the periodic boundary conditions in the $x$-direction to extend the uphill portion of the problem without increasing the number of lattice points. Thus, a problem which would have required a 720 by 218 lattice is fit into a 432 by 218 lattice. The 15,552 grains are arranged in a rectangular block of 144 by 108 and are given the same coefficient of restitution parameters as in the hourglass problem.

The results are shown in figures 3.34a and 3.34b. The momentum gained by the grains in flowing down the incline has carried them across the horizontal ground to the upper diagonal wall. Here they have formed a static pile, since they are resting on a horizontal surface and the angle of the surface of the pile is less than the angle of repose. Note also the trajectory (in red) of one of the marker particles which started out at the upper right corner of the initial block of grains. This particle bounces along the surface of the flow, following parabolic trajectories between successive contacts with the other grains (an effect known as saltation). This simulation demonstrates the ability of the lattice grain dynamics paradigm to handle a granular flow in which one of the flow boundaries is a free surface constrained by gravity and consisting of saltating grains.

*Poiseuille flow*

The final problem simulates Poiseuille flow driven by gravity using 508,032 grains in a 8064 by 128 lattice (a density of one grain for every two lattice points); the coefficient of restitution was independent of velocity and was set to .9375 for grain-grain collisions and was set to .75 for grain-wall collisions. This simulation ran for 27 consecutive hours on 512 nodes of the NCUBE CPC, and is intended as an example of the largest size problems which can be done with lattice grain dynamics on the NCUBE. The grains were initially at rest, and were accelerated to the right with a constant gravitational acceleration. The plot of the state of the simulation after 18,432 time steps is shown in figure 3.35a. A plug of grains moving with nearly uniform velocity can be seen in the center of the simulation. A narrow band (approximately 30 grain diameters wide) of highly agitated grains has

formed along each side wall, as can be seen from the color coded display of thermal velocity. The plots of average flow velocity components and thermal velocity are given in figure 3.35b, and again show the expected plug formation in the center with two narrow bands of high shear rate material near each wall. The thermal velocity is seen to reach a maximum in the high shear rate regions near each wall where it is being generated. It then decreases as it diffuses towards the walls (where it is being absorbed by the inelasticity of grain-wall collisions) and as it diffuses towards the center (where it is absorbed by the slight inelasticity of grain-grain collisions in the plug flow region). The reason for the slight asymmetry in the curve of the average $x$-component of velocity is unknown; it may be due to the fact that the simulation has not reached a steady state condition.

Plots of the shear stress and pressure on the top and bottom side walls are given in figure 3.36. Inasmuch as the grains started with no thermal velocity, the initial shear stress and pressure were zero. As the grains accelerated down the pipe, they gained thermal velocity; the shear stress and pressure increased accordingly, while maintaining a constant ratio of $\sim 0.25$. The expected shear stress for the steady state condition can be determined from the magnitude of the gravitational acceleration and the density of particles. In each time step, the mass of particles is accelerated by gravitational forces and decelerated by the shear stress forces exerted by the wall; in a steady state condition, these forces must be equal. Since the shear stress on the wall shown in figure 3.36 is expressed in terms of change in velocity per unit time per wall particle, the expected shear stress is:

$$\text{shear stress} = \frac{g \times \text{number of movable particles}}{\text{number of wall particles}}$$

For a gravitational acceleration of 8, the steady state shear stress is expected to be 252. The final shear stress was approximately 150, and thus the simulation did not reach a steady state. From figure 3.36, it can be seen that the shear stress and pressure were continuing to rise at the end of the simulation; consequently it is reasonable to expect that a steady state would have been reached if the simulation were run for a sufficient period of time.

**Figure Captions**

Figure 3.1: Illustration of the six possible velocity directions for a particle in the standard lattice gas model.

Figure 3.2: Illustrations of the collision rules for the standard lattice gas model.

a) The results of a head on collision between two particles. The two possible final states are given equal probabilities.

b) The results of a symmetric collision between three particles.

All other combinations of initial particle states remain unchanged.

Figure 3.3: Illustration of the collision of two disks in two dimensions. $\vec{c}_1$ = initial velocity of disk 1. $\vec{c}_2$ = initial velocity of disk 2. $\vec{k}$ = unit vector pointing from the center of disk 1 to the center of disk 2 at the moment of contact (the "impact parameter"). Note that because the particles are constrained to lie on the nodes of the lattice, $\vec{k}$ can have only one of six directions.

Figure 3.4: Example of the trajectory of a particle in a gravitational field. The times at which the particle has moved one lattice space from its initial position ($t_1$, $t_2$, $t_3$, and $t_4$) are shown by the intersections of its trajectory with a unit circle. The desired time step is the minimum positive time to move one lattice space ($t_2$ in this example).

Figure 3.5: Diagram defining the motion of a particle based upon its position offset. If the particle's position offset places it inside the circle, it remains on the current lattice point. If the particle's position offset places it outside the circle and within the borders shown, it will be moved to the corresponding lattice point. If the particle is moving exactly vertically (*i.e.*, the $x$-component of position offset is zero), then the particle is moved to the left lattice point if it is on an even row and is moved to the right lattice point if it is on an odd row.

Figure 3.6: Illustration showing what can happen if both components of the position offset are set to zero in a collision. Here, a stack of particles in a vertical gravitational field will be unphysically stable, since each collision (which

supports the weight of each particle) will set the horizontal component of position offset equal to zero, thus preventing the particle from ever moving horizontally and leaving the stack.

Figure 3.7: Diagram of a simple scanning pattern for the position updating process. Use of such a pattern will lead to an undesirable coupling between the scan pattern and the particle motions (see text).

Figure 3.8: One dimensional example of the scanning problem.

a) The direction of scan coincides with the direction of motion of the line of particles, with only the particle at the right end being moved.

b) The direction of scan is opposite to the direction of motion of the particles, with the result that the entire line moves one space.

c) If two particles are contending for one lattice space, which particle is moved is determined by the direction of scan.

Figure 3.9: Actual scanning pattern used for updating particle positions in the lattice grain dynamics paradigm.

Figure 3.10: Illustration of the results of a collision between two rows of particles, where the top row was initially moving vertically downward and the bottom row was initially moving vertically upward, and where the order of binary collisions with adjacent particles is the same for each lattice point.

Figure 3.11: Actual scanning pattern used for evaluating particle collisions in the lattice grain dynamics paradigm on odd time steps. The pattern for even time steps is the reverse of this.

Figure 3.12: Plot of relative rebound velocity as a function of relative incident velocity.

Figure 3.13: Examples of walls made up of wall particles:

a) a nominally flat wall (*i.e.*, a straight line of wall particles);

b) an extra rough wall.

Figure 3.14: Diagram of information exchange between a processor and its eight immediate neighbors in a concurrent processor computer. Information

to the four corner processors is not sent directly, but is sent through the left and right side processors. Similarly, information from the four corner processors travels through the top and bottom processors.

Figure 3.15: Diagram of a four-dimensional hypercube array of processors unfolded into a two-dimensional array and showing the interconnections between processors.

Figure 3.16: Plots of the initial positions and velocities of the particles for the elastic grain-gas problem.

a) The initial particle positions.

b) The initial particle velocities.

Figure 3.17: Plots of the positions and velocities of the particles for the elastic grain-gas problem after 1024 time steps.

a) The particle positions.

b) The particle velocity distributions:

1) red curve = measured distribution of the magnitude of the $x$-component of velocity,

2) green curve = measured distribution of the magnitude of the $y$-component of velocity,

3) blue curve = measured distribution of the magnitude of the velocity,

4) black curves = theoretical distributions for a two-dimensional gas.

Figure 3.18: Plot of the initial particle positions and velocities for the explosion simulation.

a) The initial particle positions. The gas consists of one particle for every three lattice points.

b) A four times blow up of the central region showing the initial velocities of the 33 by 35 rectangle of particles which represent the explosion.

Figure 3.19: Plots of the particle positions and average velocities after 128 time steps for the explosion simulation.

a) The particle positions.

b) The particle velocities averaged over 8 by 8 lattice point cells.

Figure 3.20: Plot of the initial particle positions for the Couette flow problem.

Figure 3.21: Plots of the positions and velocities for the elastic Couette flow problem after 1024 time steps.

a) The particle positions.

b) The average $x$- and $y$-components of velocity and the thermal velocity versus distance across the flow, averaged along each row.

1) red curve = average $x$-component of velocity,

2) green curve = average $y$-component of velocity,

3) blue curve = thermal velocity.

Figure 3.22: Plots of the average shear stress and pressure per wall particle versus time for the Couette flow:

1) average shear stress per wall particle on the bottom wall,

2) average pressure per wall particle on the bottom wall,

3) average shear stress per wall particle on the top wall,

4) average pressure per wall particle on the top wall.

a) The top and bottom walls are moving with a speed of 4096.

b) The top and bottom walls are moving with a speed of 16,384.

Figure 3.23: Plots of the positions and velocities for the inelastic Couette flow problem after 1024 time steps.

a) The particle positions.

b) The average $x$- and $y$-components of velocity and the thermal velocity versus distance across the flow, averaged along each row.

1) red curve = average $x$-component of velocity,

2) green curve = average $y$-component of velocity,

3) blue curve = thermal velocity.

Figure 3.24: Plots of the initial particle positions (a) and the particle positions after 2048 time steps (b) for the hourglass problem.

Figure 3.25: Diagram of a particle on the surface of the slope of a two-dimensional stack of particles showing the potential barrier which must be overcome for it to "escape."

Figure 3.26: Plot of the number of particles which have flowed through the hourglass as a function of time.

Figure 3.27: Diagram of the experimental apparatus from Nedderman, Davies, and Horton (1980).

Figure 3.28: Plot of the initial particle positions for the flow around a circular obstacle problem. The initial plots for the other obstacles are similar.

Figure 3.29: Plots of the particle positions after 512 time steps (a), the paths of the marker particles after 1536 time steps (b), and the experimentally observed streamline patterns (c) for the flow around a circular obstacle.

Figure 3.30: Plots of the particle positions after 512 time steps (a), the paths of the marker particles after 1536 time steps (b), and the experimentally observed streamline patterns (c) for the flow around a square obstacle.

Figure 3.31: Plots of the particle positions after 512 time steps (a), the paths of the marker particles after 1536 time steps (b), and the experimentally observed streamline patterns (c) for the flow around a triangular obstacle oriented point up.

Figure 3.32: Plots of the particle positions after 512 time steps (a), the paths of the marker particles after 1536 time steps (b), and the experimentally observed streamline patterns (c) for the flow around a triangular obstacle oriented point down.

Figure 3.33: Plot of the initial particle positions for the flow down an inclined slope. Gravitational acceleration is oriented so that the diagonal line is horizontal and the wall across the bottom of the problem is at an angle of 23.4 degrees to the horizontal.

Figure 3.34: Plots of the particle positions (a) and the paths of the marker particles (b) for the flow down an inclined slope after 6144 time steps.

Figure 3.35: Plots of the results of the Poiseuille flow problem after 18,432 time steps.

a) A plot of the average particle velocities (averaged over rectangular 224 by 3 areas of the lattice) with the background color of an area determined by the thermal velocity in that area.

b) Plots of the average $x$- and $y$-components of velocity and the thermal velocity versus distance across the flow, averaged along each row.

1) average $x$-component of velocity,

2) average $y$-component of velocity,

3) thermal velocity.

Figure 3.36: Plots of the average shear stress and pressure per wall particle versus time for the Poiseuille flow:

1) average shear stress per wall particle on the bottom wall,

2) average pressure per wall particle on the bottom wall,

3) average shear stress per wall particle on the top wall,

4) average pressure per wall particle on the top wall.

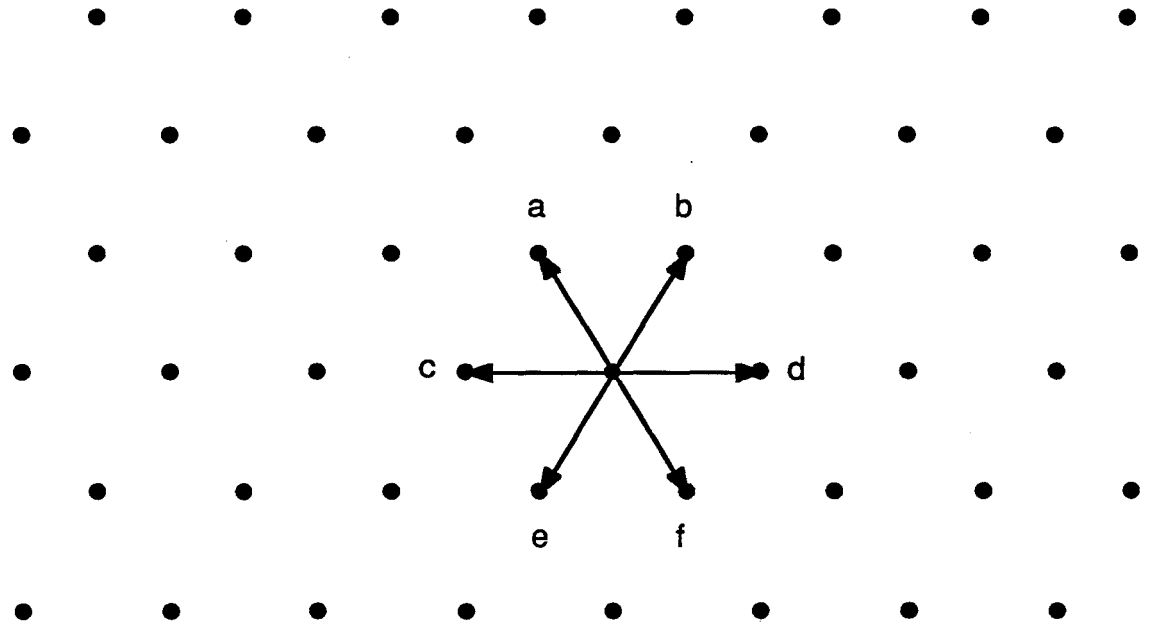



Figure 3.1

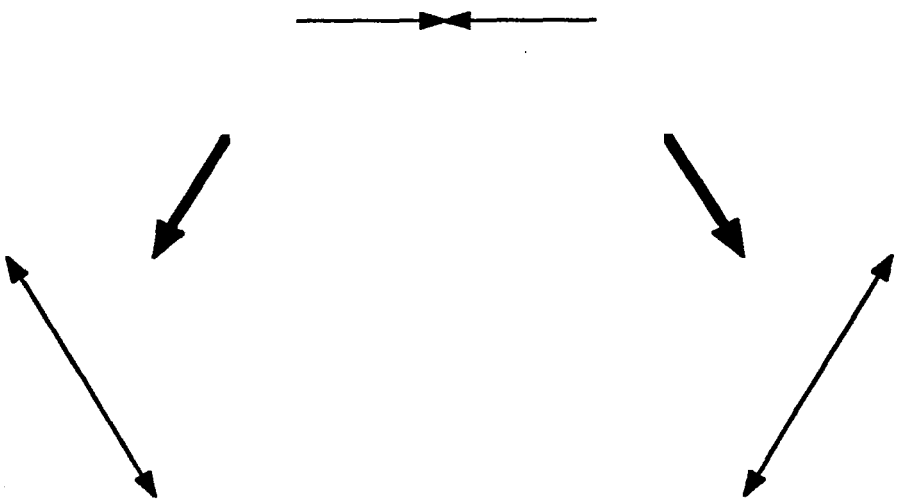


Figure 3.2a

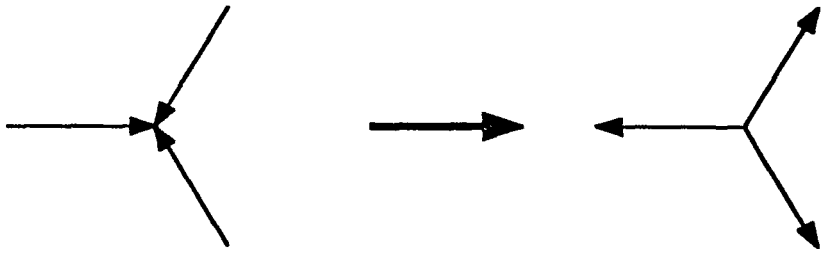


Figure 3.2b

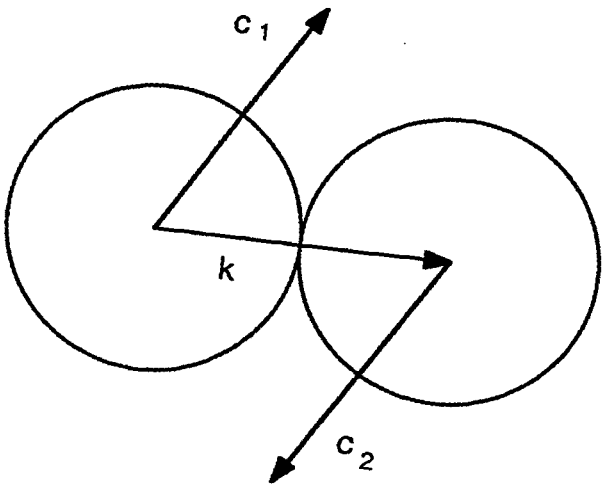



Figure 3.3

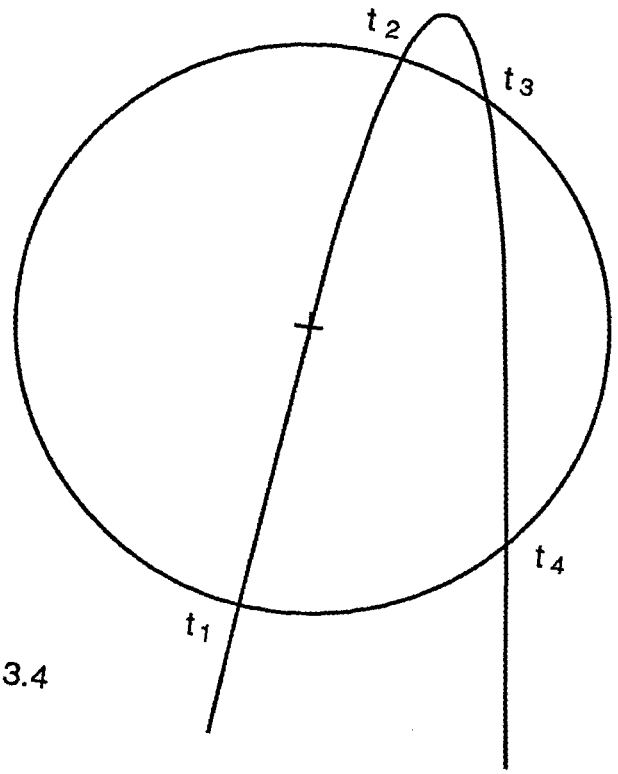



Figure 3.4

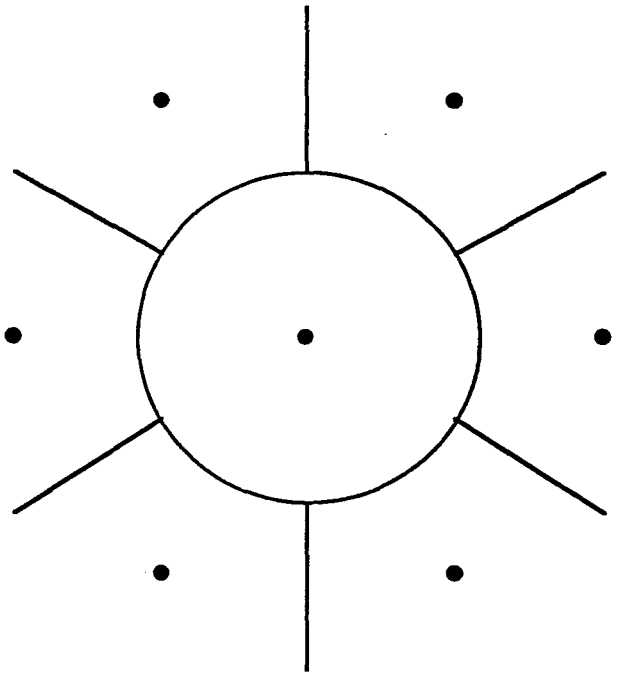


Figure 3.5

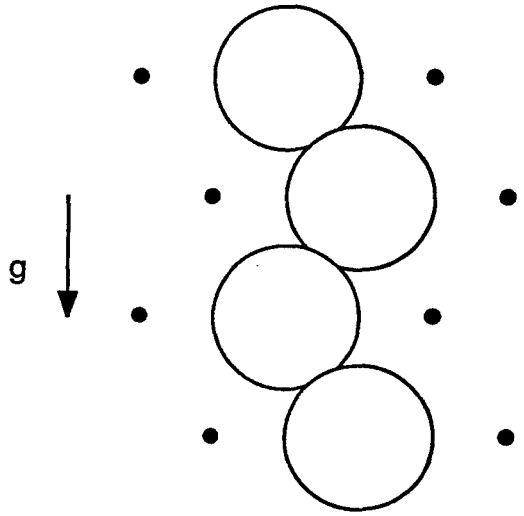



Figure 3.6

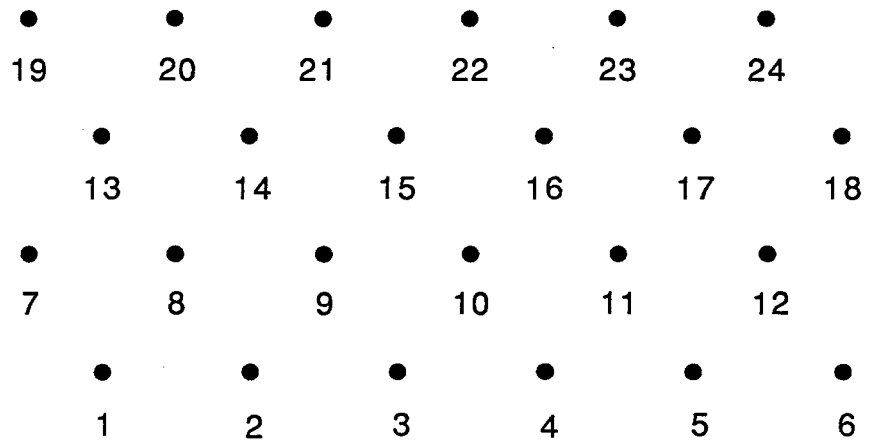



Figure 3.7

Figure 3.8a

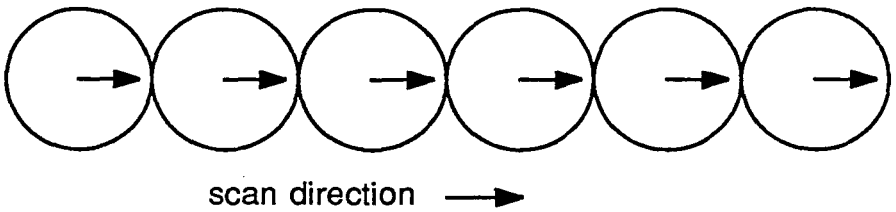
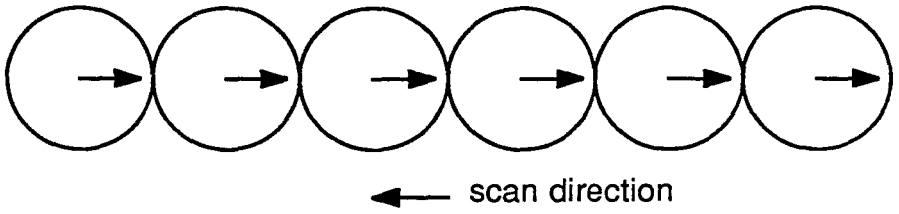



Figure 3.8b



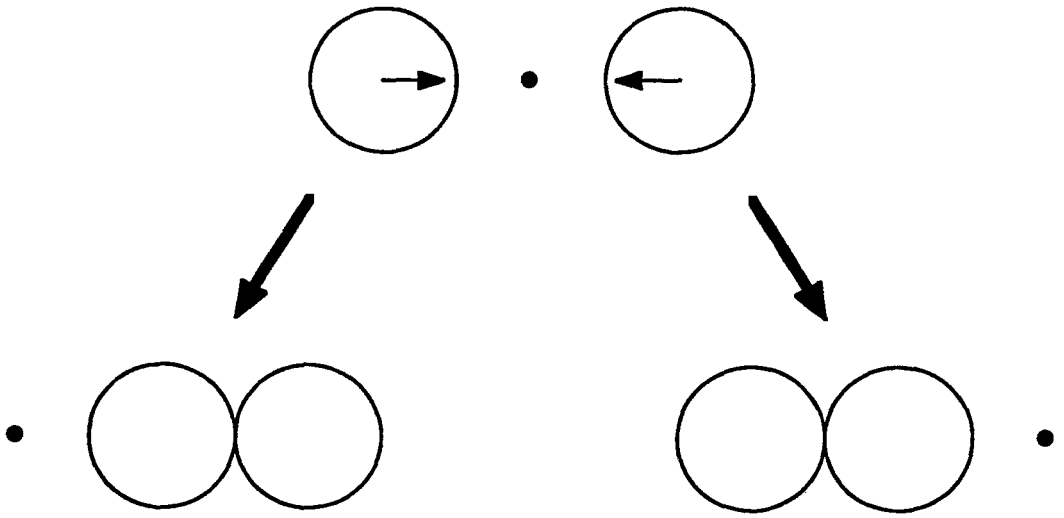



Figure 3.8c

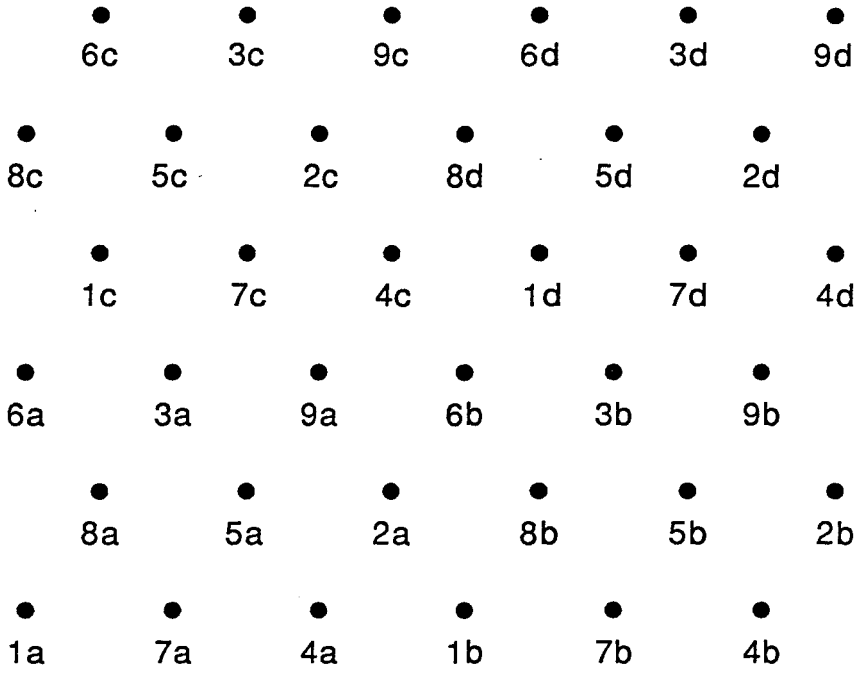



Figure 3.9

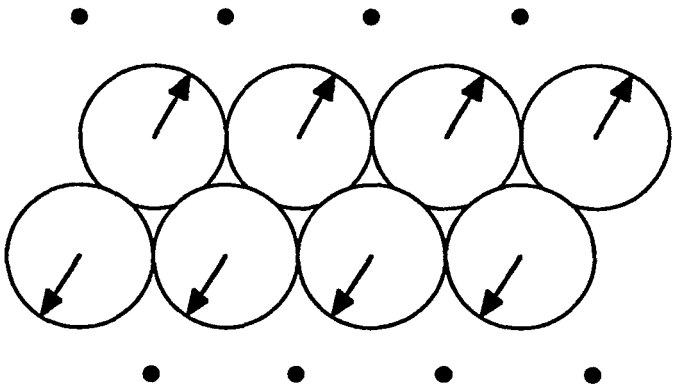


Figure 3.10

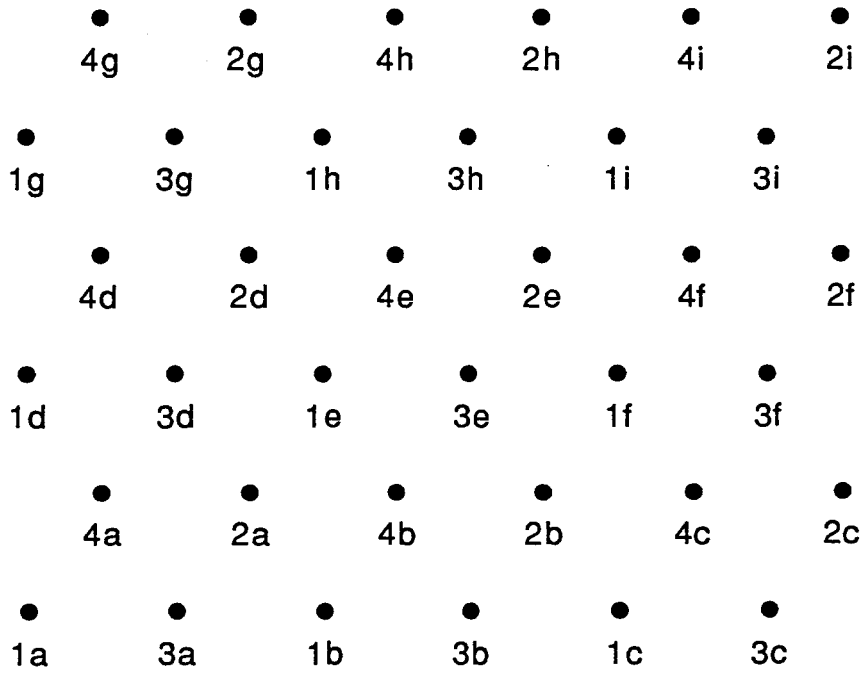



Figure 3.11

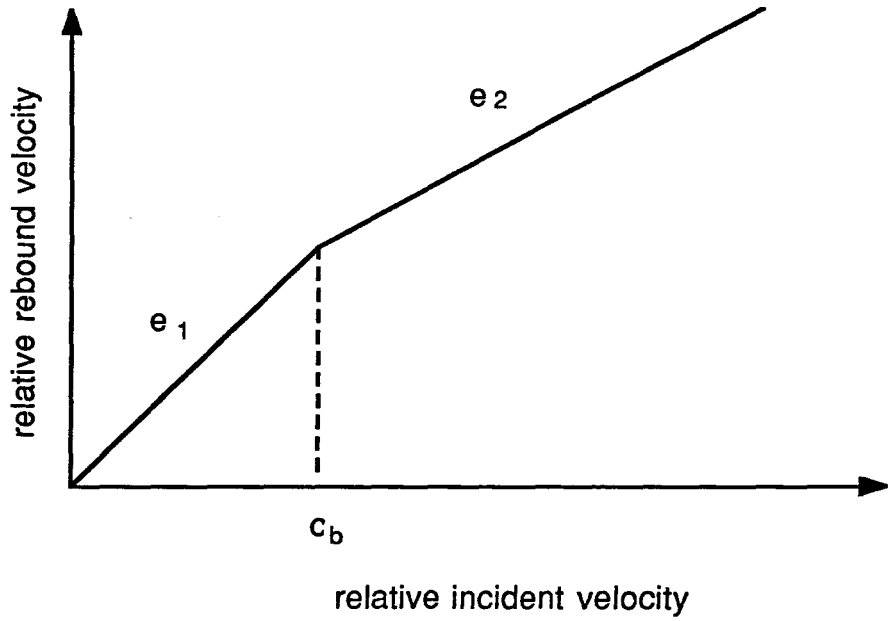



Figure 3.12

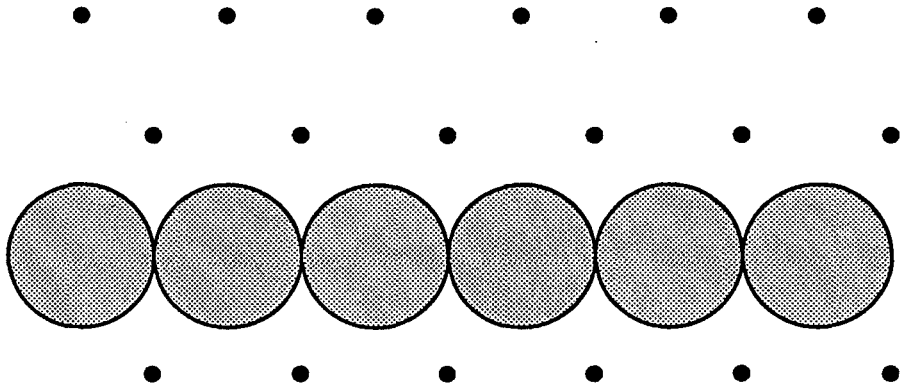


Figure 3.13a

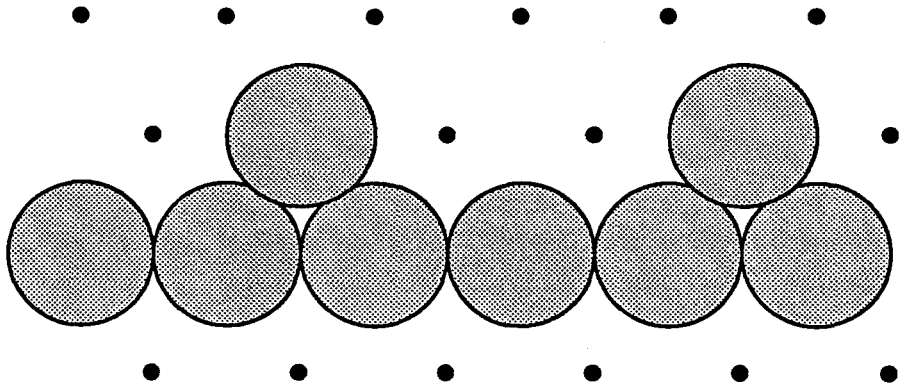


Figure 3.13b

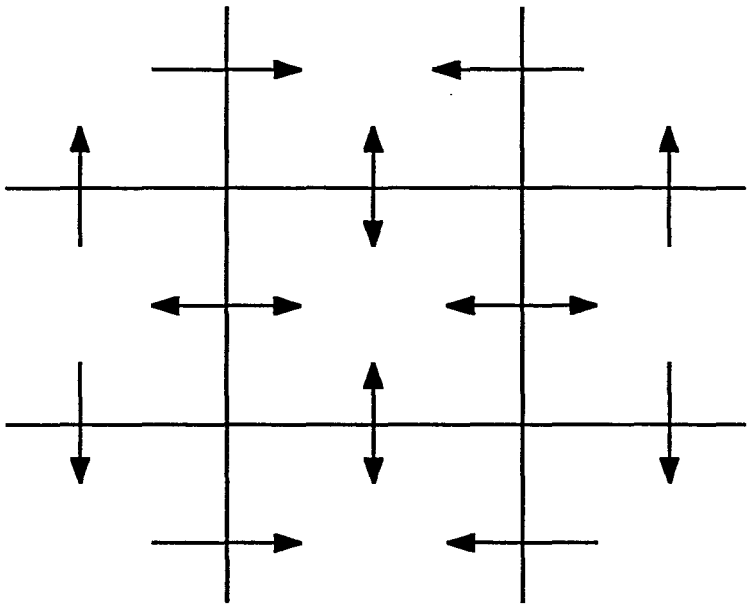


Figure 3.14

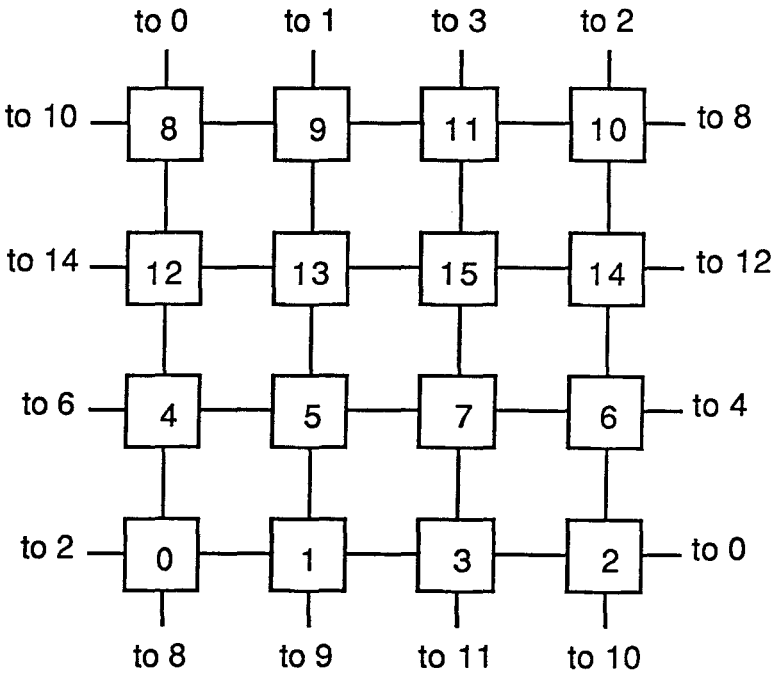



Figure 3.15

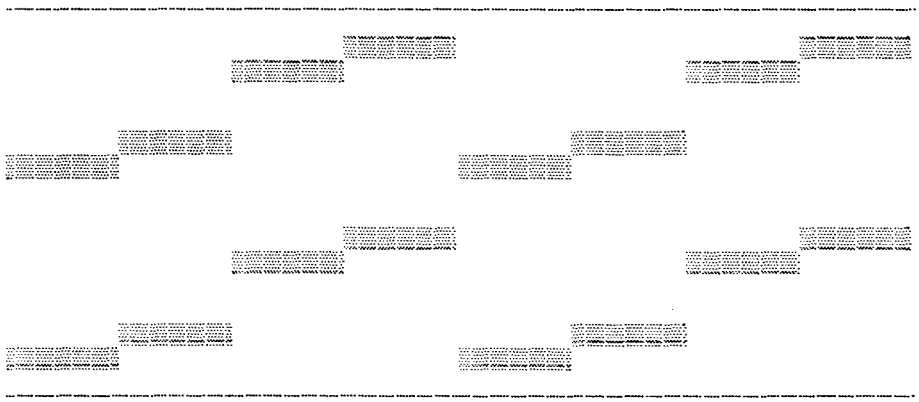


Figure 3.16a: The initial particle positions for the grain-gas problem

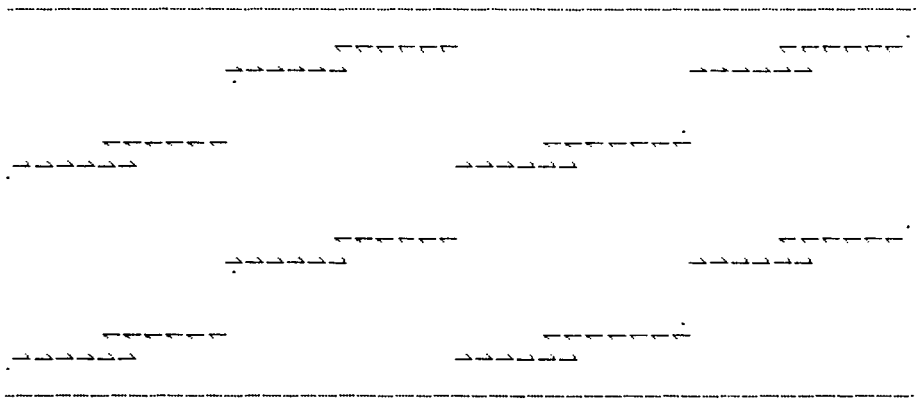


Figure 3.16b: The initial particle velocities for the grain-gas problem

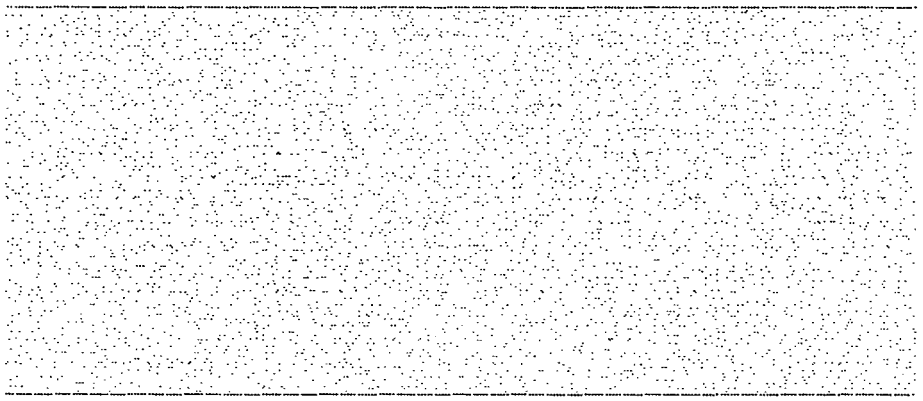


Figure 3.17a: The particle positions after 1024 time steps for the grain-gas problem

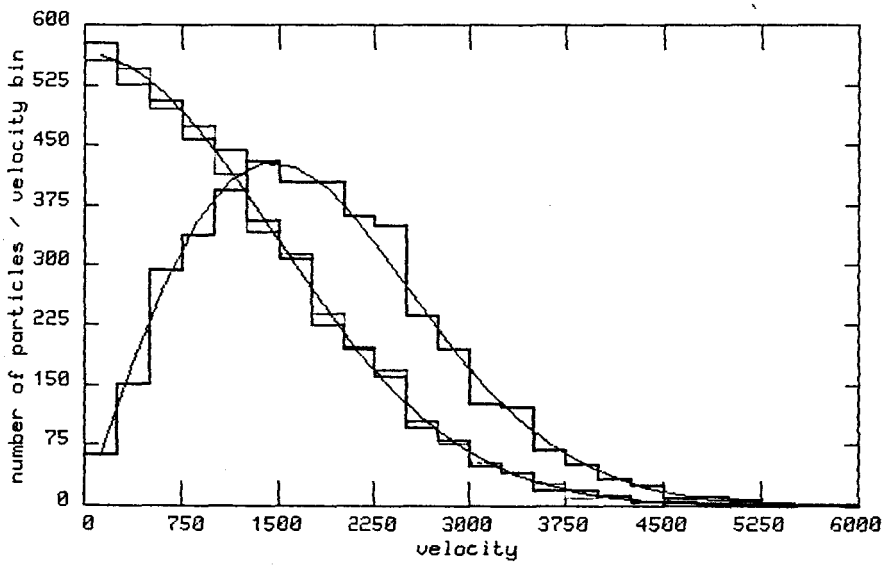



Figure 3.17b: The particle velocity distributions for the grain-gas problem

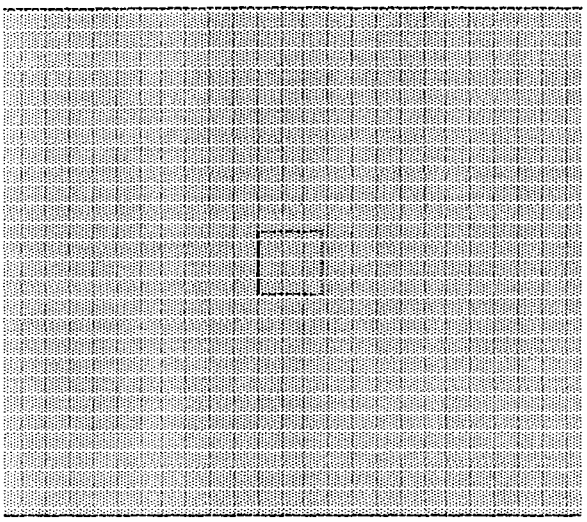


Figure 3.18a: The initial particle positions for the explosion simulation.

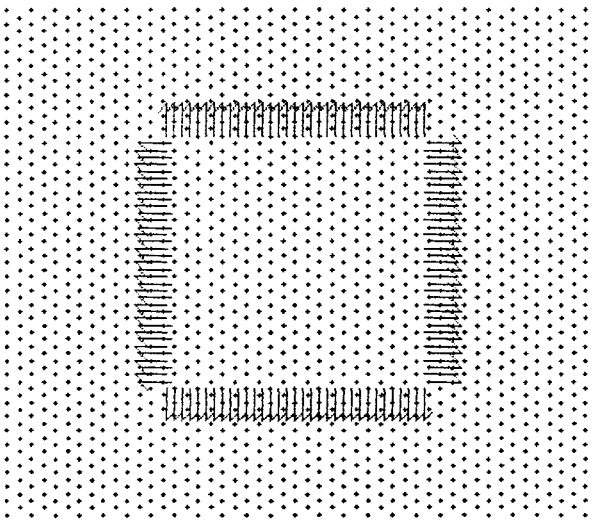


Figure 3.18b: A four times blow up of the central region showing the initial velocities.

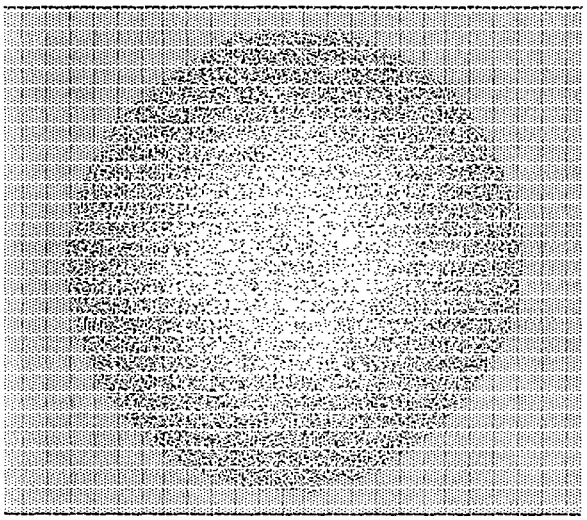


Figure 3.19a: The particle positions after 128 time steps for the explosion simulation.

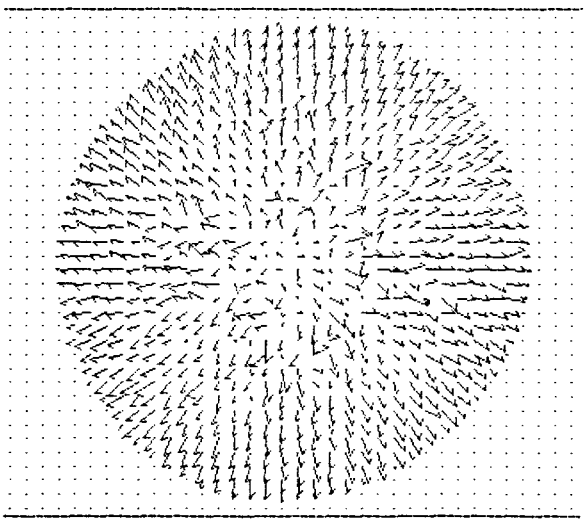


Figure 3.19b: The particle velocities after 128 time steps for the explosion simulation.

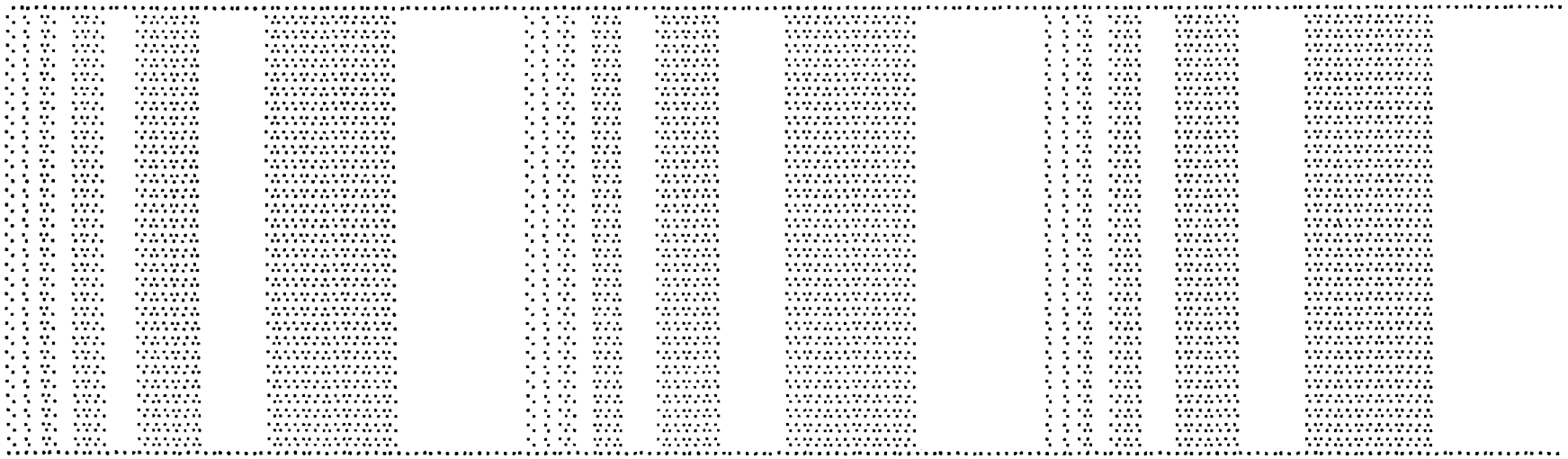


Figure 3.20: Plot of the initial particle positions for the Couette flow problem.

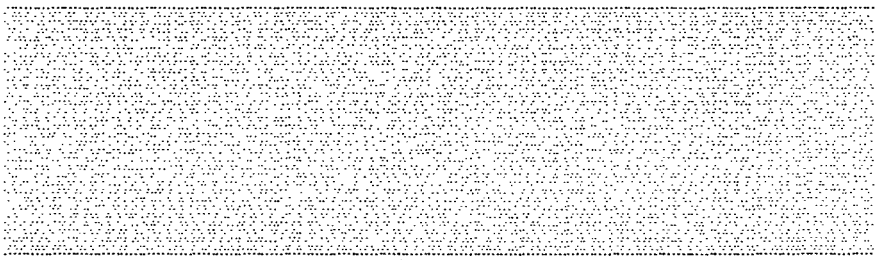


Figure 3.21a: The particle positions after 1024 time steps for the elastic Couette flow problem.

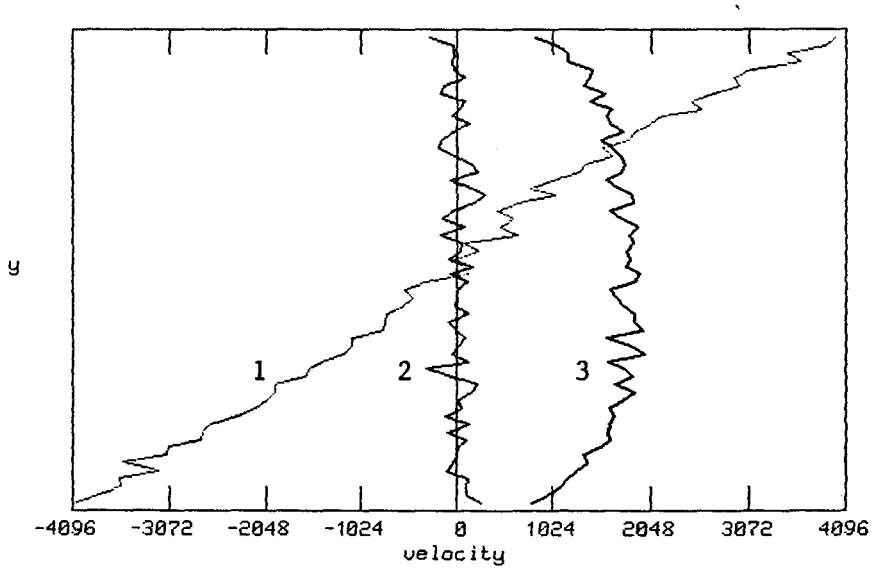



Figure 3.21b: The particle velocities after 1024 time steps for the elastic Couette flow problem.

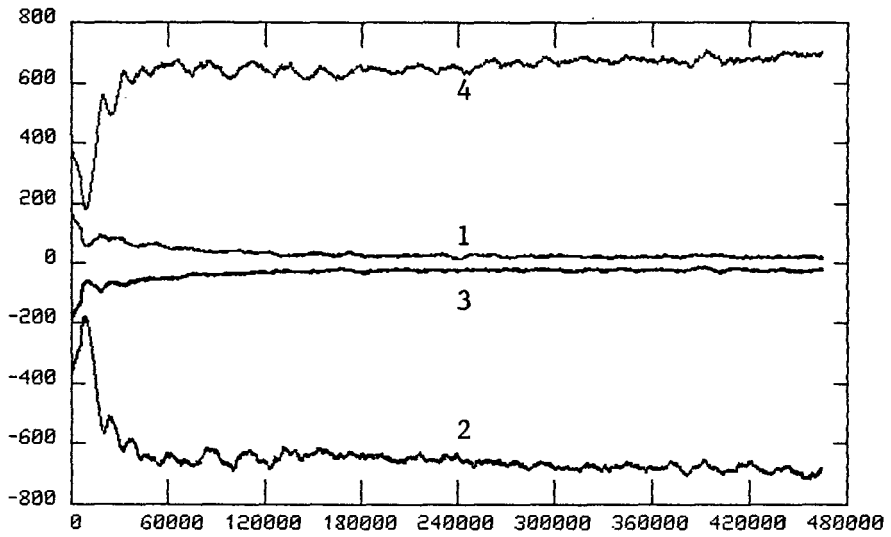



Figure 3.22a: Plots of the average shear stress and pressure per wall particle versus time for the elastic Couette flow problem.

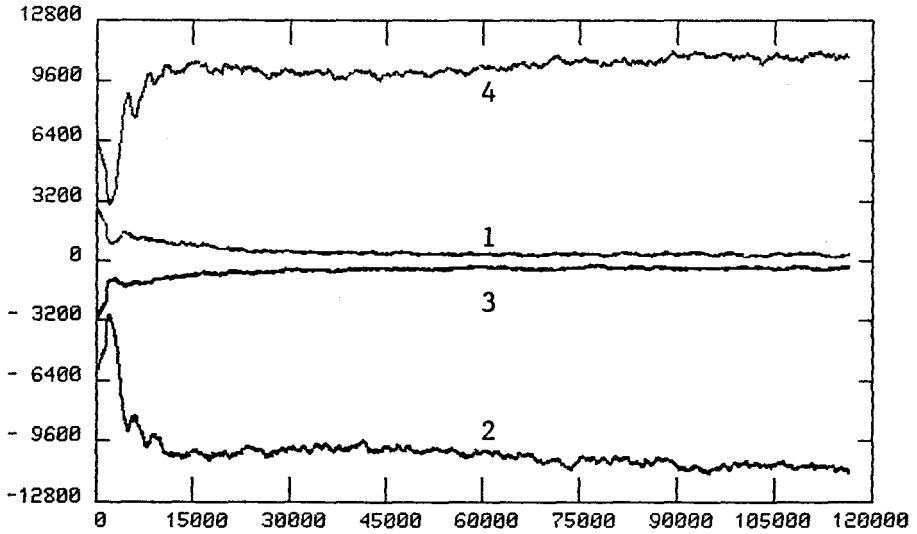



Figure 3.22b: Plots of the average shear stress and pressure per wall particle versus time for the elastic Couette flow problem.

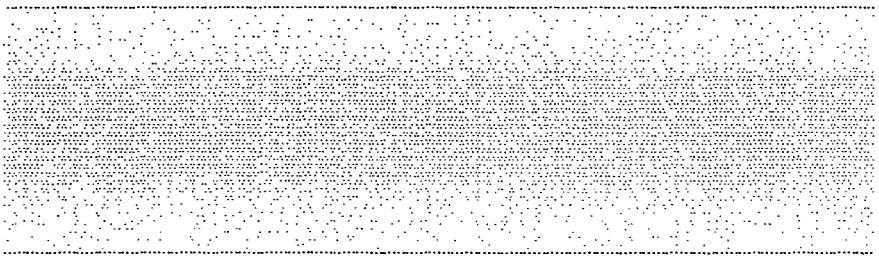


Figure 3.23a: The particle positions after 1024 time steps for the inelastic Couette flow problem.

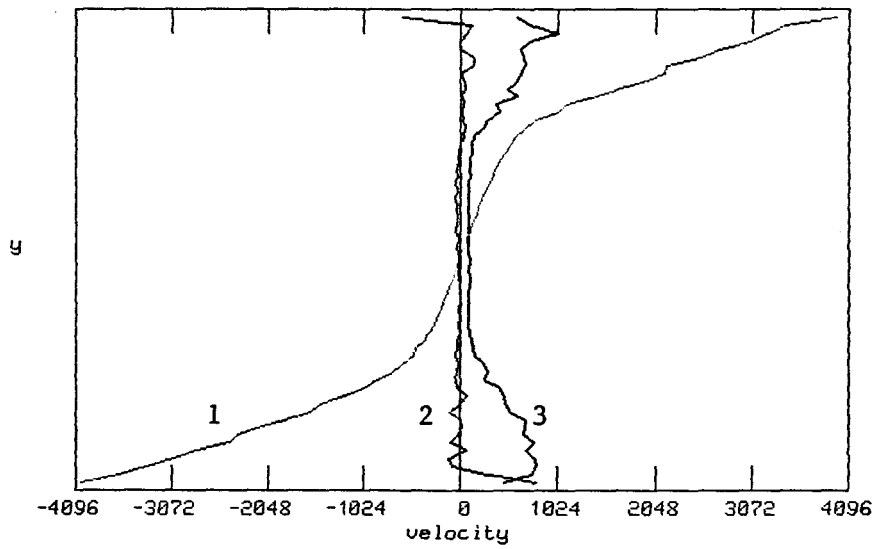



Figure 3.23b: The particle velocities after 1024 time steps for the inelastic Couette flow problem.

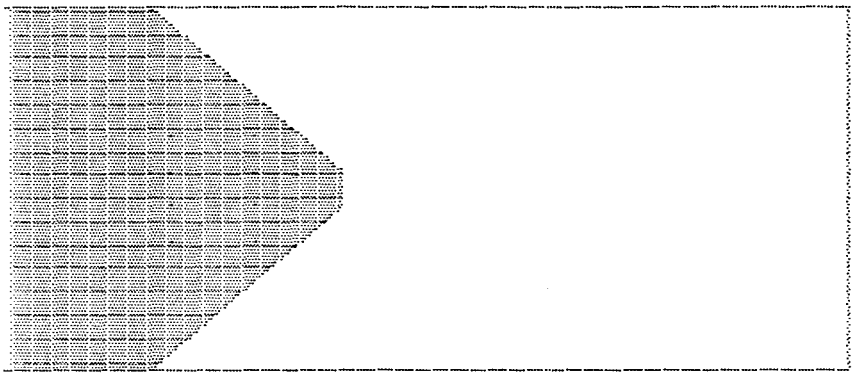


Figure 3.24a: The initial particle positions for the hourglass problem.

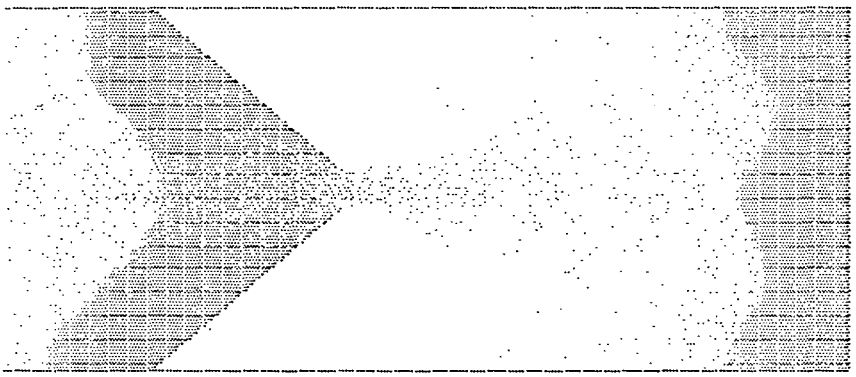


Figure 3.24b: The particle positions after 2048 time steps for the hourglass problem.

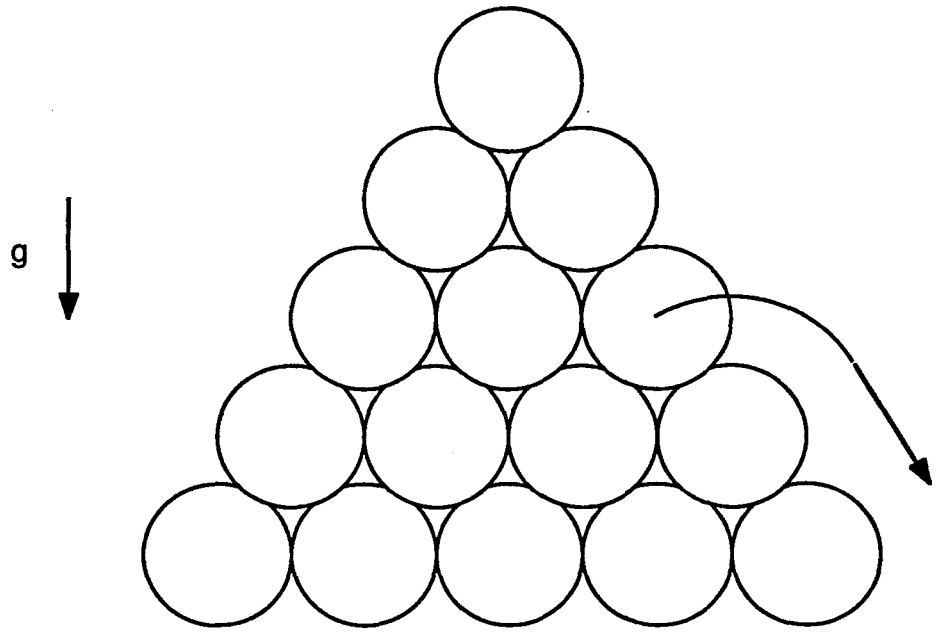



Figure 3.25

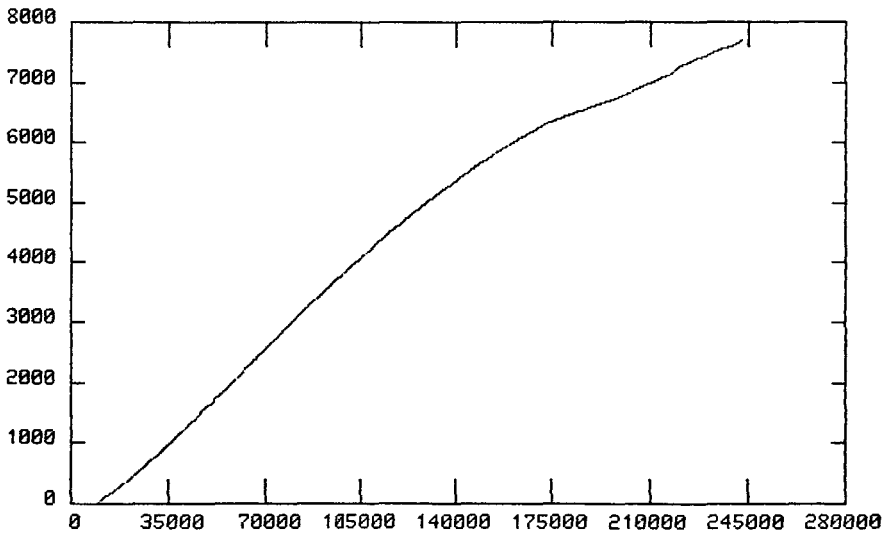



Figure 3.26: The number of particles which have flowed through the hourglass as a function of time.

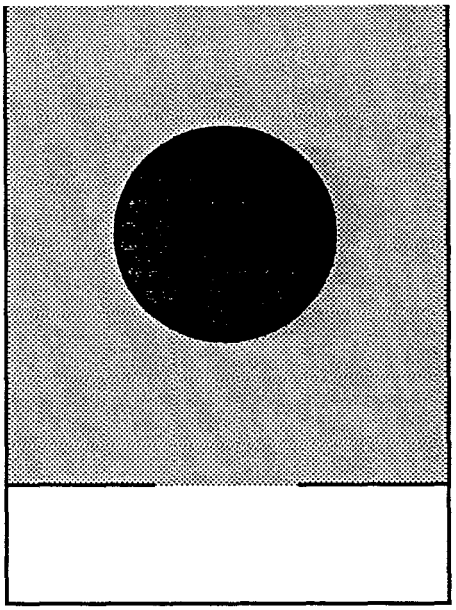


Figure 3.27

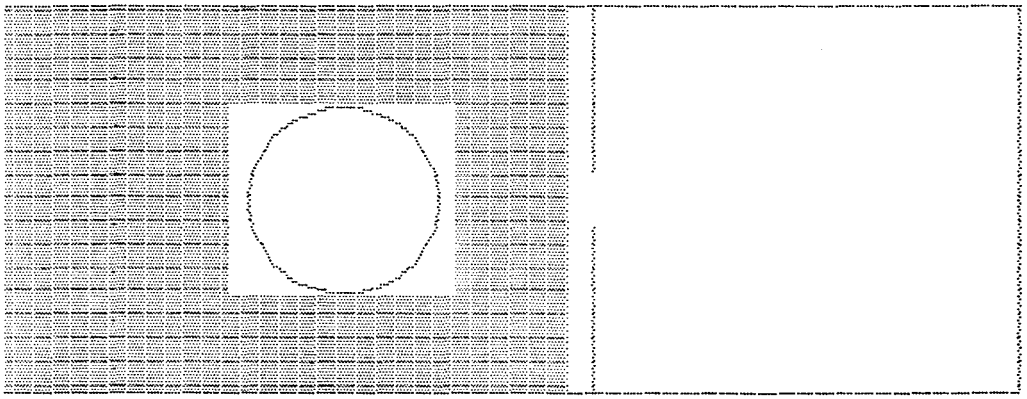


Figure 3.28: The initial particle positions for the flow around a circular obstacle.

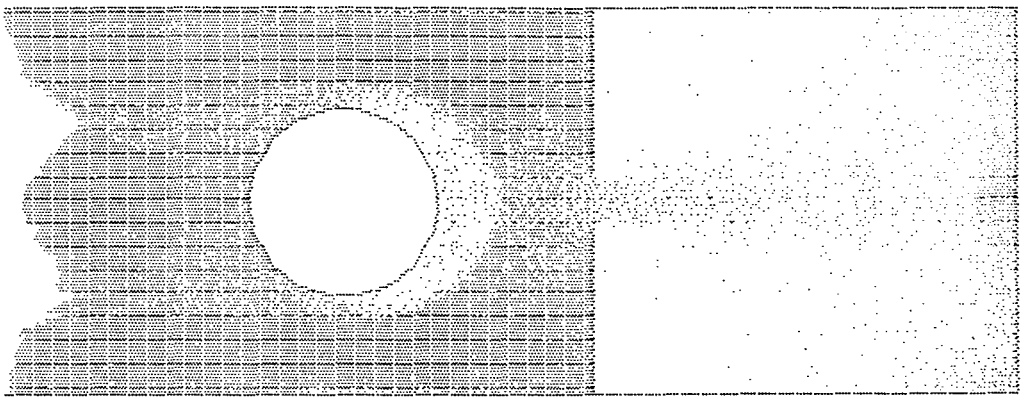


Figure 3.29a: The particle positions after 512 time steps for the flow around a circular obstacle.

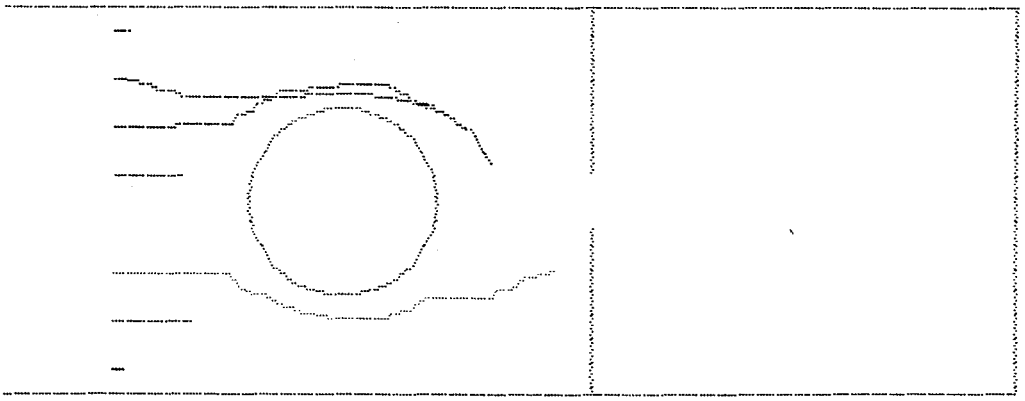


Figure 3.29b: The paths of the marker particles after 1536 time steps for the flow around a circular obstacle.

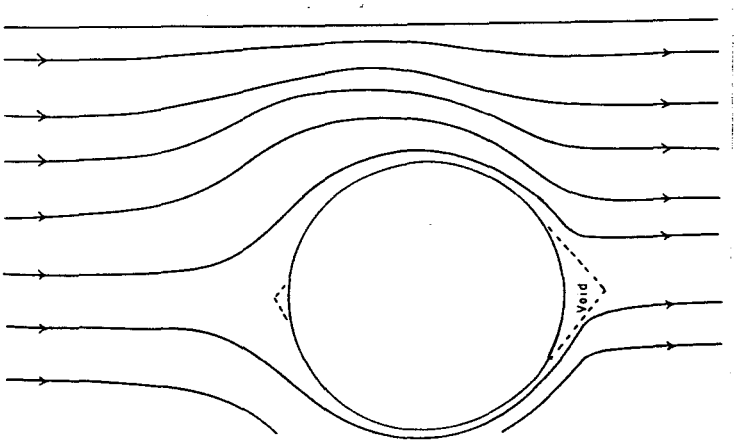



Figure 3.29c: The experimentally observed streamline patterns for the flow around a circular obstacle.

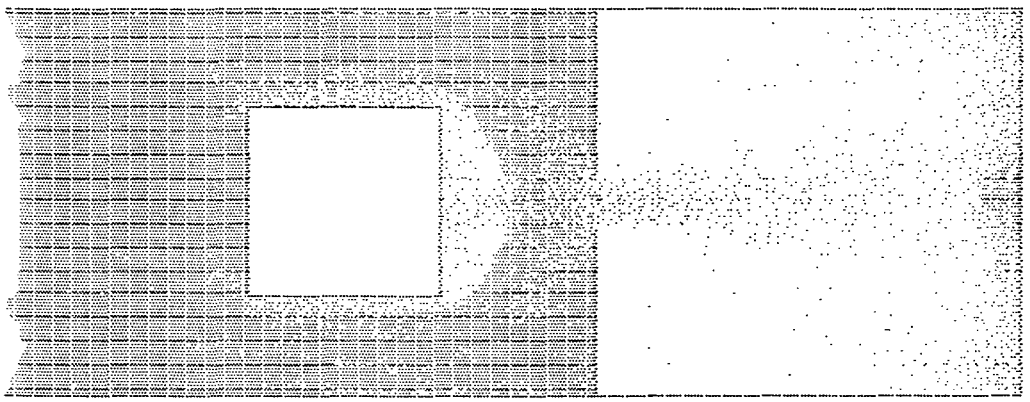


Figure 3.30a: The particle positions after 512 time steps for the flow around a square obstacle.

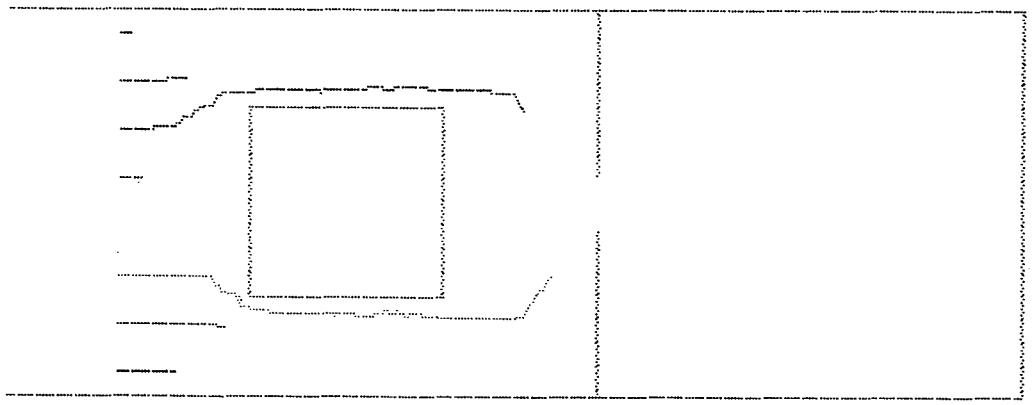


Figure 3.30b: The paths of the marker particles after 1536 time steps for the flow around a square obstacle.

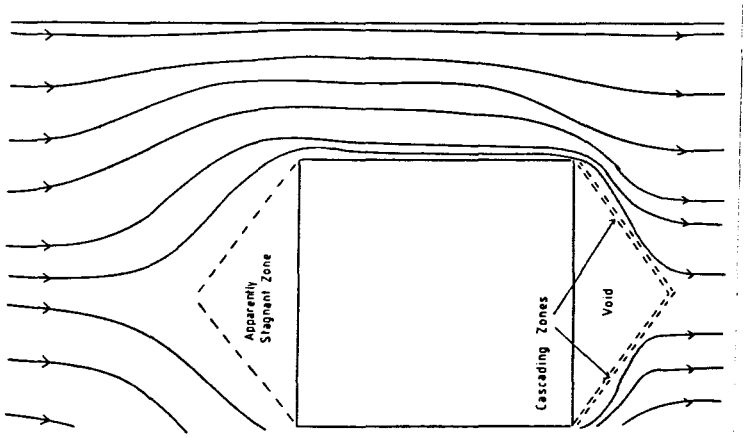



Figure 3.30c: The experimentally observed streamline patterns for the flow around a square obstacle.

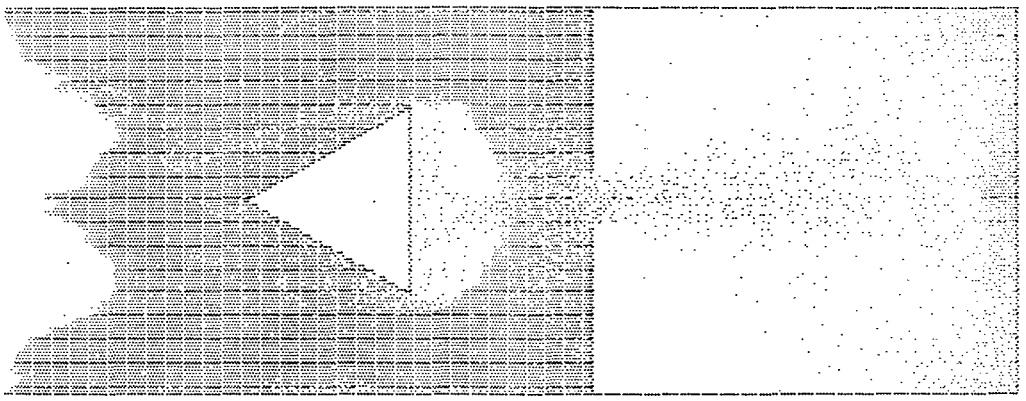


Figure 3.31a: The particle positions after 512 time steps for the flow around a triangular obstacle.

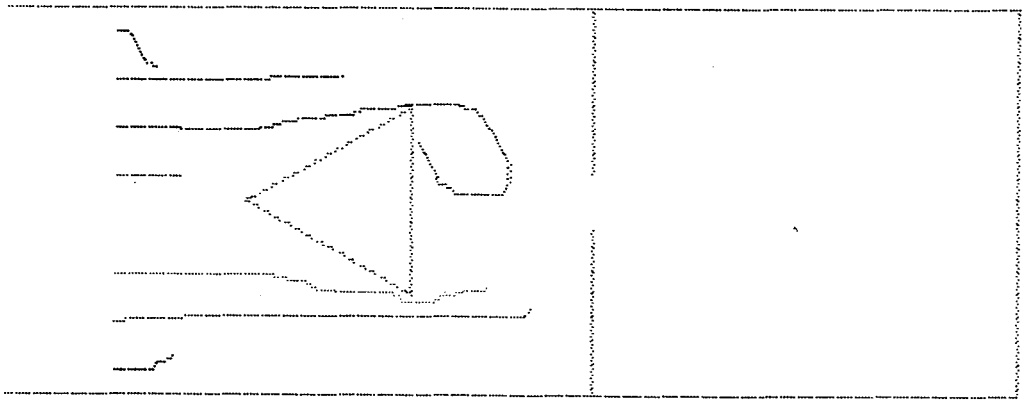


Figure 3.31b: The paths of the marker particles after 1536 time steps for the flow around a triangular obstacle.

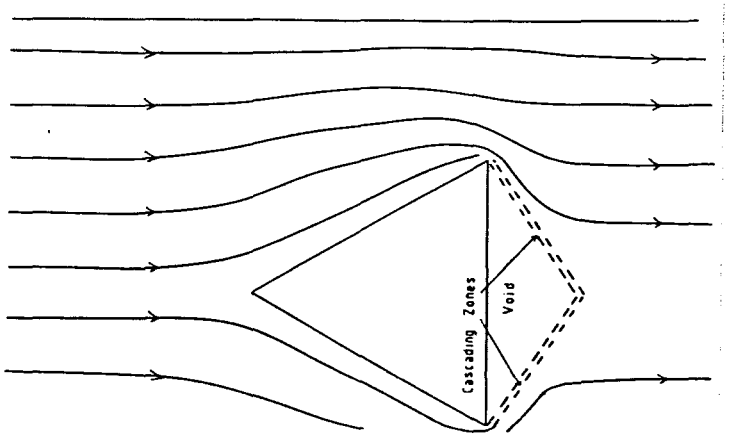



Figure 3.31c: The experimentally observed streamline patterns for the flow around a triangular obstacle.

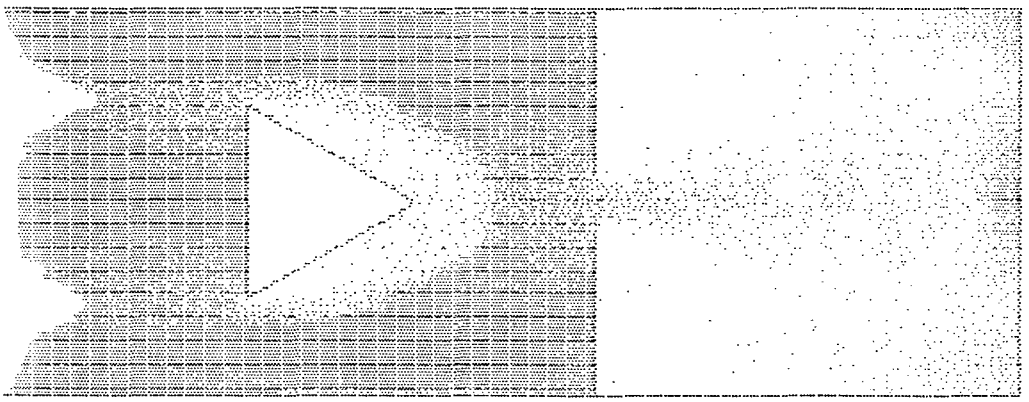


Figure 3.32a: The particle positions after 512 time steps for the flow around a triangular obstacle.

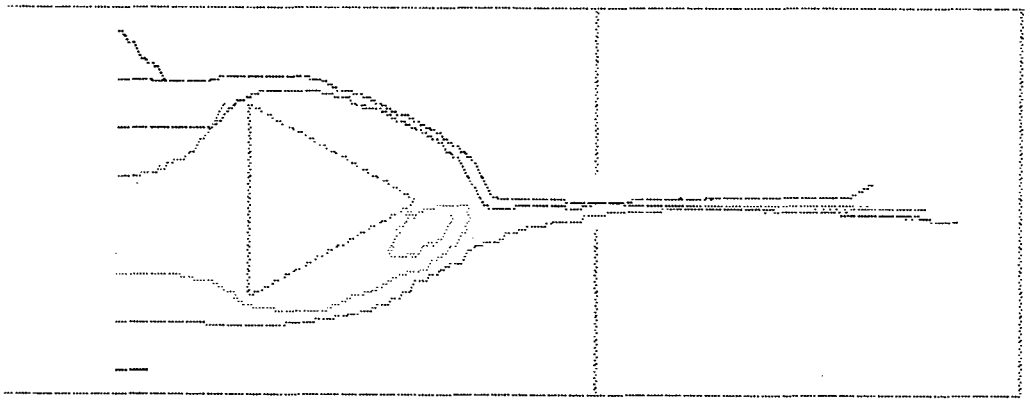


Figure 3.32b: The paths of the marker particles after 1536 time steps for the flow around a triangular obstacle.

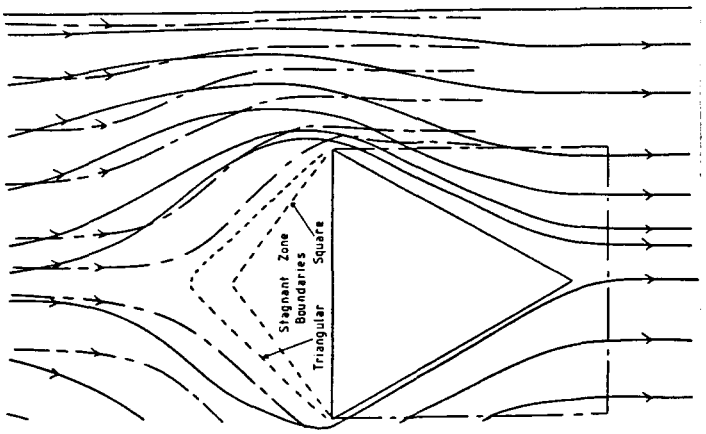



Figure 3.32c: The experimentally observed streamline patterns for the flow around a triangular obstacle.

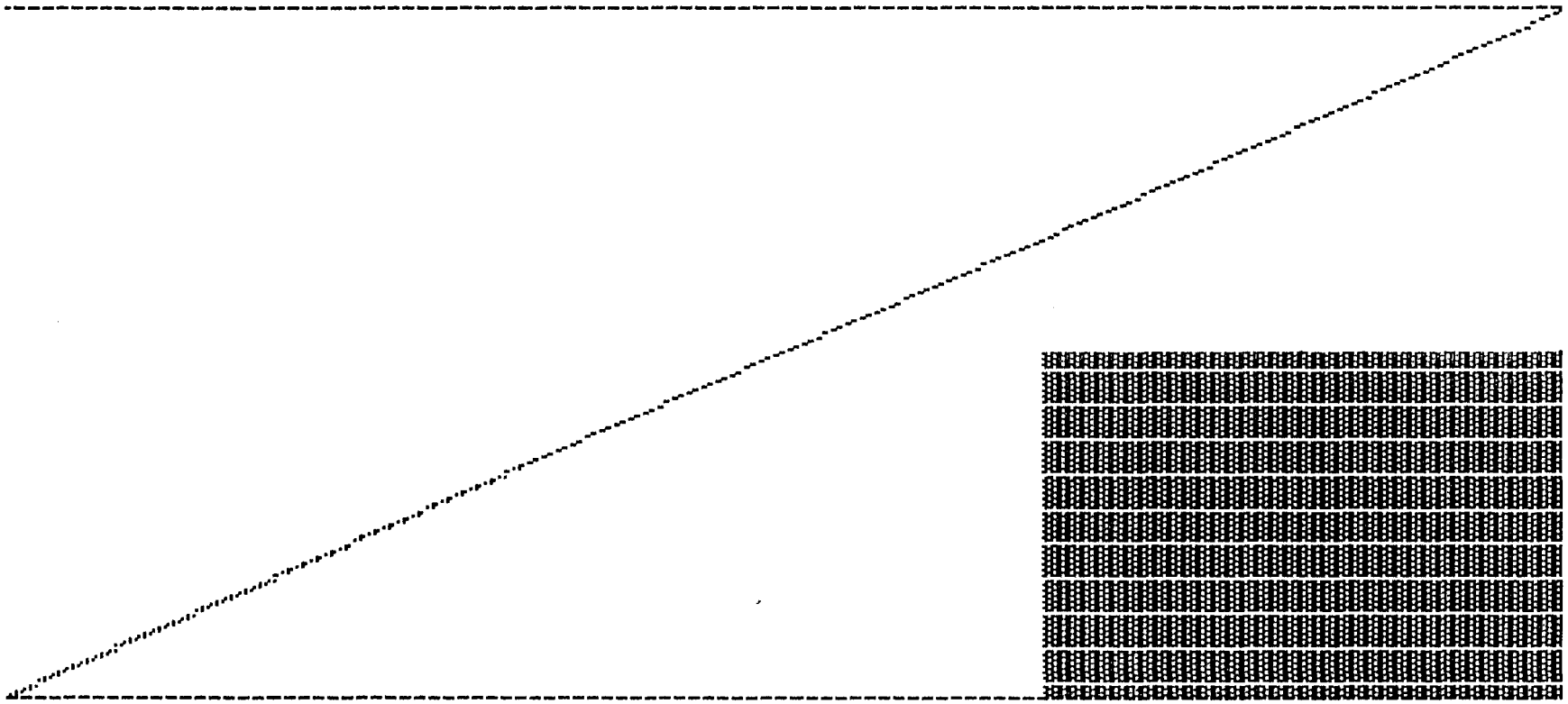


Figure 3.33: The initial particle positions for the flow down an inclined slope.

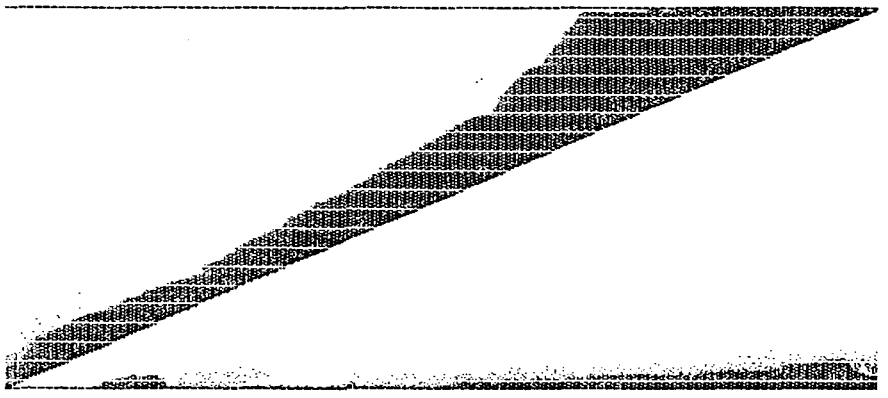


Figure 3.34a: The particle positions after 6144 time steps for the flow down an inclined slope.

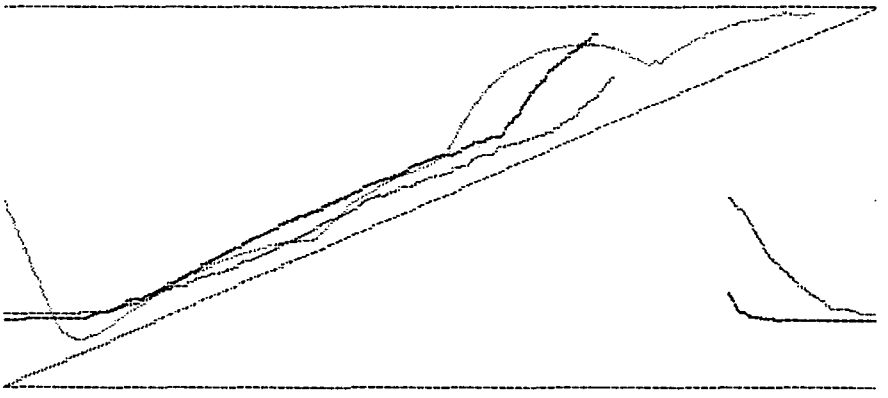


Figure 3.34b: The paths of the marker particles after 6144 time steps for the flow down an inclined slope.

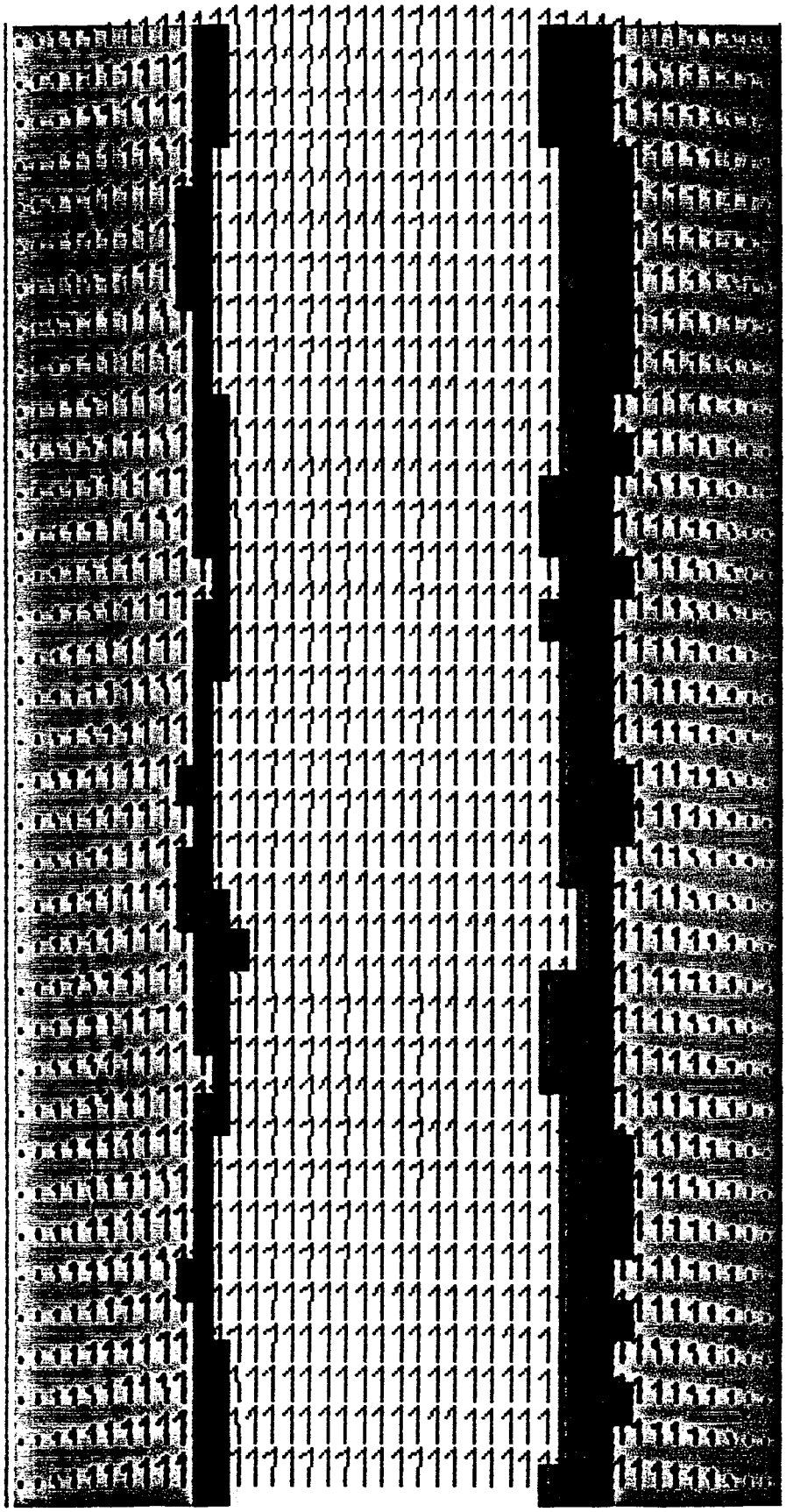


Figure 3.35a: The average particle velocities and thermal velocities for the Poiseuille flow problem.

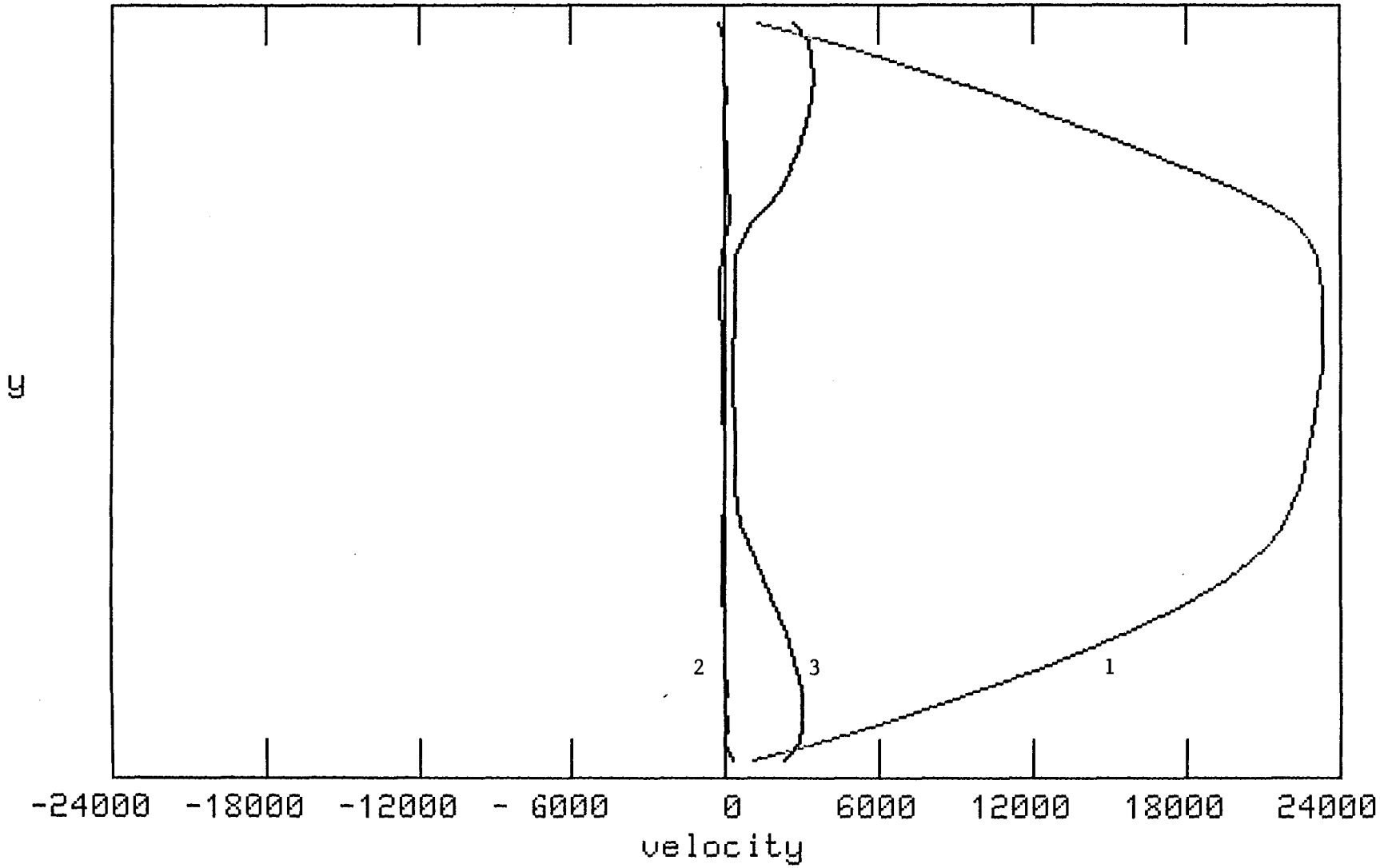



Figure 3.35b: The average velocity components and thermal velocity for the Poiseuille flow.

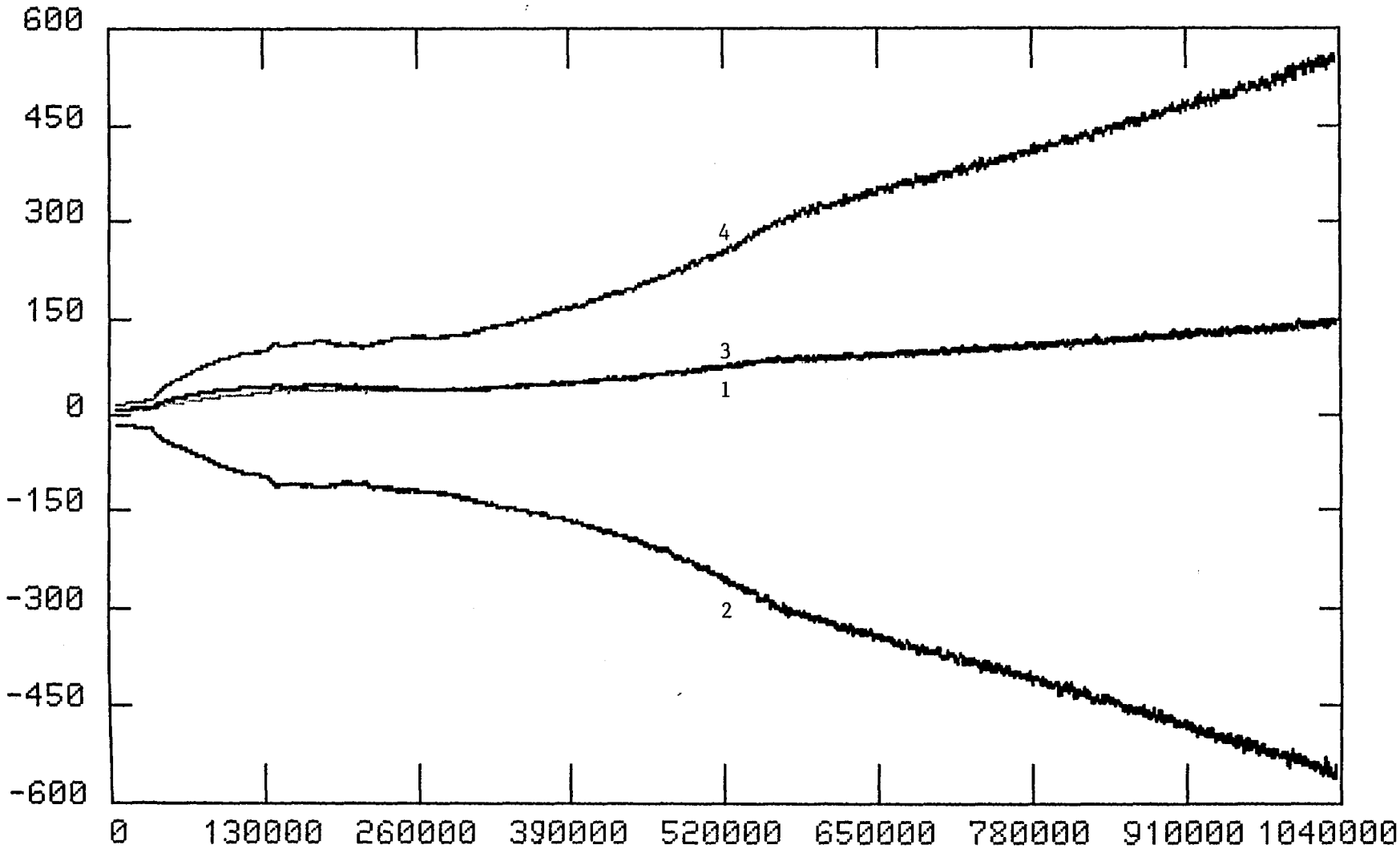



Figure 3.36: The average shear stress and pressure per wall particle versus time for the Poiseuille flow.

# Conclusions

There seems to be an unstated assumption in the education of a physicist that the field of classical physics is a closed book; all that needs to be known about classical physics has been discovered. In recent years, it has gradually become clear that this assumption is unjustified, partly due to the discoveries in the fields of classical chaos and complex systems. One such complex classical system which has received only intermittent attention is the granular system. Despite the fact that granular systems are a fairly ubiquitous phenomena in our world, there exists relatively little understanding of them, due in large part to the difficulty of experimentally observing their internal workings and the difficulty of finding an adequate theoretical framework for describing them. In this study, we have examined three aspects of the theoretical description of granular systems in an attempt to better understand their inner workings, and to better predict their external forms.

In Chapter 1, we have taken an existing continuum theory for the flow of smooth, non-spinning grains and extended it to cover the case of rough, spinning grains. It was shown that a self-consistent continuum theory for spinning grains could be derived by using techniques from the field of nonequilibrium thermodynamics. This theory was based on the idea of describing the flow using the average translational flow velocity, the thermal translational velocity, the average spin velocity, and the thermal spin velocity of the grains. These quantities were then related to each other and to the properties of the individual grains through the momentum equations, the energy equations, and the constitutive relations. Several illustrative examples were presented that showed that energy and momentum could be transferred between the spin and translational modes. Also, it was demonstrated that the addition of spin and surface roughness to the theory resulted in higher values of the shear stress to pressure ratio and the average translational flow velocity to thermal translational velocity ratio at the wall. Future work in this area should extend this theory to include non-spherical particles, where the transfer of energy and momentum in a collision between particles depends more on geometric effects than on surface roughness.

In Chapter 2, we have taken a careful look at the question of boundary conditions at a wall on a simple continuum theory of granular flow. The traditional, simple boundary conditions of fluid mechanics were seen to be inappropriate for use in granular systems. Instead, a theory based on the idea of three separate slips at the wall (average flow velocity slip, thermal velocity slip, and density slip) was derived; and equations relating these slips to the particle properties, the wall properties, and the bulk flow variables were obtained. The example of Couette flow was presented, and parametric studies of the variation in flow characteristics as a function of particle and wall parameters were done. Now that the principles of various slips at a boundary have been established, the next extension of these boundary conditions would be to include the effects of surface roughness and spin of the particles.

It was argued in Chapter 3 that perhaps a differential equation formulation of granular material dynamics was not the best approach. As an alternative, a theory based on the use of cellular automata was presented (the lattice grain dynamics paradigm). This theory used a highly simplified model of grain-grain interactions with grain positions confined to an underlying lattice in order to allow the simulation of many thousands of grains. Since the calculations involved in these simulations were of a distributed nature, it was possible to program these simulations on a concurrent processor computer. Examples were shown of large scale granular flows including flows around obstacles, flows down inclined slopes, and flows through pipes. Although the lattice grain dynamics paradigm is reasonably successful, several improvements and extensions are desirable. The first of these involves the treatment of static stresses and pressures in piles of grains in a gravitational field. In its current form, the paradigm handles these by allowing the particles to "vibrate" while remaining at a fixed position in the pile, thus transmitting the appropriate stresses and pressures through collisions. Although this results in the correct stresses and pressures, the individual particle velocities are no longer necessarily an accurate representation of the particles' true velocities. It may be worthwhile to simply pay the price of additional storage and computation time needed to store and update

the static contact forces between adjacent particles in a compact mass. Other more sophisticated improvements include the incorporation of particle spin effects and the extension to three dimensions. Hopefully, it will someday be possible to use a technique like this to conduct "experiments" on granular systems entirely within a computer and to display the results on a real time basis.

## Appendix 1. The Size of the Time Step

The time step in the lattice grain dynamics paradigm is set to the time required for the fastest particle to travel one lattice spacing. For the case of zero gravitational acceleration, this is simply given by the lattice spacing divided by the speed of the fastest particle. For the case of non-zero gravitational acceleration, the fastest particle's trajectory may cross the one lattice spacing boundary several times (figure 3.4); here, the desired time step size is the minimum positive time needed to travel one lattice space. The times at which the fastest particle's trajectory crosses the one lattice spacing boundary are given by the solutions to:

$$d^2 = \frac{1}{4}g^2t^4 + (\vec{g}\cdot\vec{v})t^3 + v^2t^2$$

where:

$d$ = distance between lattice points,

$g$ = gravitational acceleration magnitude,

$\vec{g}$ = gravitational acceleration vector,

$t$ = time,

$v$ = velocity magnitude of the fastest particle,

$\vec{v}$ = velocity vector of the fastest particle.

Since an analytical solution to this fourth order polynomial equation is not available, a numerical solution is required.

The numerical solution technique is based upon two considerations: a) we are only interested in the smallest positive solution; and b) this smallest positive solution will generally be only slightly greater than the time required to travel one lattice space when the gravitational acceleration lies along the velocity vector of the particle (*i.e.*, $\vec{g}\cdot\vec{v} = gv$). This minimum positive time is given by:

$$t_{min} = \frac{2d}{v + \sqrt{v^2 + 2gd}}$$

The numerical solution starts with this minimum time (rounded down to the nearest integer), and increments it in unit steps, comparing the total distance travelled after each step to one lattice spacing. The integer time at which the particle is closest to travelling one lattice space is then taken to be the desired time step.

## Appendix 2. The Smooth Disk Collision Model

The particles in the lattice grain dynamics paradigm are assumed to be smooth, circular disks which are constrained to lie on the vertices of a two-dimensional, triangular lattice. Their collisions are assumed to be instantaneous, and the results of a collision may be derived from conservation of momentum, from a model of rebound velocity as a function of incident velocity, and from the lack of surface friction.

The model of rebound velocity in a collision allows for the loss of kinetic energy due to inelasticity by using a velocity dependent coefficient of restitution. The slope of the relative rebound velocity versus relative incident velocity curve has one value ($e_1$) up to a given incident velocity ($c_b$), and has a separately defined value ($e_2$) above that velocity (figure 3.12).

Thus, the equations of constraint for the collision of two disks are:

1. Conservation of momentum:

$$m_1(\vec{c}_1' - \vec{c}_1) = m_2(\vec{c}_2' - \vec{c}_2).$$

2. Relative rebound velocity:

$$\vec{k} \cdot \vec{c}_{12}' = \begin{cases} -e_2(\vec{k} \cdot \vec{c}_{12} - c_b) - e_1 c_b, & \text{if } c_{12} > c_b; \\ -e_1(\vec{k} \cdot \vec{c}_{12}), & \text{otherwise.} \end{cases}$$

3. Conservation of angular momentum with no surface friction:

$$\vec{k} \times \vec{c}_{12}' = \vec{k} \times \vec{c}_{12}.$$

where:

$\vec{c}_1$ = velocity of particle 1,

$\vec{c}_2$ = velocity of particle 2,

$\vec{c}_{12} = \vec{c}_1 - \vec{c}_2$ (relative velocity),

$\vec{k}$ = unit vector defined in figure 3.3,

$m_1$ = mass of particle 1,

$m_2$ = mass of particle 2,

primed values = values after the collision,

unprimed values = values before the collision.

From these equations we obtain the velocity of particle 1 after the collision:

$$\vec{c}_1' = \begin{cases} \vec{c}_1 - \left[\dfrac{(1+e_2)(\vec{k}\cdot\vec{c}_{12}) + (e_1 - e_2)c_b}{1 + m_1/m_2}\right]\vec{k}, & \text{if } c_{12} > c_b; \\ \vec{c}_1 - \left[\dfrac{(1+e_1)(\vec{k}\cdot\vec{c}_{12})}{1 + m_1/m_2}\right]\vec{k}, & \text{otherwise.} \end{cases}$$

Note that for the case of two moving particles of equal mass, the term in the denominator $(1 + m_1/m_2)$ is equal to 2; while for the case of a moving particle (particle 1) hitting a wall particle (particle 2) this denominator will equal 1, since a wall particle is regarded as having infinite mass. This and the fact that $e_1$ and $e_2$ are separately defined for particle-particle and particle-wall collisions are the only differences between the two types of collisions.

## Appendix 3. Lattice Grain Dynamics Program Listing

```
/*  27 sep 88        gmg
    This program simulates granular systems using lattice grain dynamics
    techniques, with periodic boundary conditions in the x-direction,
    on the NCUBE hypercube concurrent computer under cubix/crosIII.
    It includes the position offsets of the particles and
    a velocity dependent coefficient of restitution, and
    calculates the movements and collisions in a complex pattern.
*/
#include "cros.h"
#include <stdio.h>
extern double sqrt();
int id, icb, idl, idx, idy, iep1, iep2, iew1, iew2;          /* parameter file */
int igx, igy, jx1, jx2, kq, kx, ky, ke;                      /* parameter file */
int mvx, mvy, nt, nto, nxt, nyt, ntu;                        /* parameter file */
int icp, icw, id2, ke1, ke2, kxp, kxw, kqp, kqw, m0, m1, m2;
int n3, n4, nb[2], ndy1, np, npx, nx, ny0, ny1, ny2;
struct lattis { long ipx, ipy, ivx, ivy; short mlp, nlp;};
/*  ipx = x component of position offset
    ipy = y component of position offset
    ivx = x component of particle velocity
    ivy = y component of particle velocity
    mlp = mode of lattice point
        = -1 for a wall particle
        =  0 for an empty lattice point
        =  1 for a movable particle which may be moved
        =  2 for a movable particle which does not need to be moved
    nlp = number of marked particle
        =  0 for unmarked particle
*/
    struct lattis la[4489];
struct lsindx { long ni, no, pi, po;};
    struct lsindx lx, ly;
struct pkvlct { long ipv2, ipvx, ipvy;};
unsigned int mskn[2], mskp[2];
```

```
main()
{
    char nam1[8], nam2[8], nam3[8], nam4[8], nam5[8];
    double av[4], dl, dl3, dp, dy1, dy3, gm, gm2, gv, gt[2];
    double pv1, pv2, rng, rt, vtt[3];
    FILE *fi, *fm, *fp, *fq;
    int md, ipx1, ipx2, ipy1, ipy2, ivxi, ivyi;          /* initial value file */
    int iox, ipv, ist, it, itm, itt, iv2p, ix, iy, ll;
    int ndx1, ndy2, nl, nn, nnp, npl, npx2, nx2, nx3, nxt2, nxt5, ny;
    int coor[2], ia[4], igt[2], nd0[2], nd1[2], nd2[2], nde[2], ndo[2], nnd[2];
    int dadd(), iadd(), imax();
    float aivt[64][48];
    long aivx[64][48], aivy[64][48], amlp[64][48];
    long idp[2], ip1, ip2, ip3, is1, is2, is3, ng, nm, nq, nw;
    struct pkvlct pv;
    struct cubenv env;
    cparam(&env);
/*
    read in the input parameters
*/
    if ((fi = fopen ("in", "r")) == 0) exit(1);
/*      fi = 0, the file cannot be opened */
    fscanf (fi, "%d\n%s\n%s\n%s\n", &ist, nam1, nam3, nam4);
/*  ist = 0 for numerical output of data for each particle at the end
        = 1 for output of average velocity vectors of groups of particles
        = 2 for numerical output of data for each particle at the end
            and numerical output of data for marked particles for each time step
        = 3 for output of average velocity vectors of groups of particles
            and numerical output of data for marked particles for each time step
        = 4 for numerical output of data for marked particles for each time step
*/
    if ((fp = fopen (nam1, "r")) == 0) exit(2);
/*      fp = 0, the file cannot be opened */
    fscanf (fp, "nam2 =%6s%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d\n",
        nam2,  &id, &icb, &iep1, &iew1);
    fscanf (fp, "%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d\n",
```

```
        &idl, &idx, &idy, &iep2, &iew2);
    fscanf (fp, "%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d\n",
              &igx, &igy, &jx1, &jx2);
    fscanf (fp, "%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d\n",
         &kq,  &kx,  &ky,  &ke);
    fscanf (fp, "%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%6d%*s%*s%6d\n",
         &mvx, &mvy);
    fscanf (fp, "%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d%*s%*s%6d\n",
         &nt, &nxt, &nyt, &ntu, &nto);
/*  nam2 = name of file containing initial data
    id   = minimum calculated displacement**2 for a particle to move
    icb  = relative collision velocity below which iep1 or iew1 apply
    idl  = lattice spacing (cm * 4096)
    idx  = change in x component of offset per vertical lattice point (cm*4096)
    idy  = change in y component of offset per vertical lattice point (cm*4096)
    iep1 = ke * ep1, ep1 = coeff. of restitution (particle-particle) below icb
    iep2 = ke * ep2, ep2 = coeff. of restitution (particle-particle) above icb
    iew1 = ke * ew1, ew1 = coeff. of restitution (particle-wall) below icb
    iew2 = ke * ew2, ew2 = coeff. of restitution (particle-wall) above icb
    igx  = x component of gravitational acceleration (cm/sec**2)
    igy  = y component of gravitational acceleration (cm/sec**2)
    jx1  = x component of velocity of lower wall
    jx2  = x component of velocity of upper wall
    kq   = denominator of components of unit vector k
    kx   = numerator of x component of unit vector k (kq * 1/2)
    ky   = numerator of y component of unit vector k (kq * sqrt(3)/2)
    ke   = constant in expressions for iep1, iep2, iew1, and iew2
    mvx  = number of points in the x direction for each average velocity vector
    mvy  = number of points in the y direction for each average velocity vector
    nt   = number of times to iterate time step loop
    nto  = number of time steps between output dumps
    ntu  = number of times particle positions are updated per time step
    nxt  = number of points in the x direction
    nyt  = number of points in the y direction
*/
    fclose (fp);
```

```
    if (nto <= 0) nto = nt;
/*      nto <= 0, then output particle data only at end of simulation */
    if (nxt % mvx != 0)
/*      nxt is not an integral multiple of mvx */
    {   printf("mvx does not divide evenly into nxt\n");
        exit(3);
    }
    if (nxt / mvx > 64)
/*      there are too many average velocity cells in the x direction */
    {   printf("nxt / mvx > 64\n");
        exit(4);
    }
    if ((nyt - 2) % mvy != 0)
/*      nyt - 2 is not an integral multiple of mvy */
    {   printf("mvy does not divide evenly into nyt - 2\n");
        exit(5);
    }
    if ((nyt - 2) / mvy > 48)
/*      there are too many average velocity cells in the y direction */
    {   printf("(nyt - 2) / mvy > 48\n");
        exit(6);
    }
    if (nxt % 6 != 0)
/*      nxt is not an integral multiple of 6 */
    {   printf("nxt must be an integral multiple of 6\n");
        exit(7);
    }
/*
    initialize various things
*/
    gm = sqrt((double)(igx * igx + igy * igy));
    gm2 = gm * gm;
    icp = icb * (iep1 - iep2);
    icw = icb * (iew1 - iew2);
    id2 = idx * 2;
/*  id2  = change in x component of offset per horizontal lattice point */
```

```
    iep1 += ke;
    iew1 += ke;
    if (gm > 0.)
/*      calculate the maximum time step for problems with gravity */
        itm = (int)(1024. * sqrt(2. * (double)idl / gm) + 0.5);
/*  itm = time for a grain with 0 initial velocity to fall 1 lattice space */
    itt = 0;
/*  itt = total elapsed time */
    ke1 = ke + ke;
    ke2 = ke / 2;
    kxp = ke * kx * 2;
    kxw = ke * kx;
    kqp = ke * kq * 2;
    kqw = ke * kq;
    nnd[0] = 1 << ((env.doc + 1) / 2);
/*  nnd[0] = number of nodes in the x direction */
    if (nnd[0] > nxt)
/*      there are more nodes than points in the x direction */
    {   printf("There are more nodes than points in the x direction\n");
        exit(8);
    }
    nnd[1] = 1 << (env.doc / 2);
/*  nnd[1] = number of nodes in the y direction */
    if (nnd[1] > nyt)
/*      there are more nodes than points in the y direction */
    {   printf("There are more nodes than points in the y direction\n");
        exit(9);
    }
    gridinit (2, nnd);
    gridcoord(env.procnum, coor);
/*  coor[0] = x coordinate of this node */
/*  coor[1] = y coordinate of this node */
    mskn[0] = gridmask (env.procnum, 0, -1);
/*  mskn[0] = mask of the channel in the negative x direction */
    mskp[0] = gridmask (env.procnum, 0,  1);
/*  mskp[0] = mask of the channel in the positive x direction */
```

```
    mskn[1] = coor[1] ==          0 ? 0 : gridmask (env.procnum, 1, -1);
/*  mskn[1] = mask of the channel in the negative y direction */
    mskp[1] = coor[1] == nnd[1] - 1 ? 0 : gridmask (env.procnum, 1,  1);
/*  mskp[1] = mask of the channel in the positive y direction */
    nx = nxt / nnd[0];
    if (coor[0] < nxt - nx * nnd[0])
/*      nxt is not a multiple of nnd[0], so add an extra column to this node */
    {   nx++;
        ndx1 = nx * coor[0] + 1;
    }
    else
        ndx1 = nxt - nx * (nnd[0] - coor[0]) + 1;
/*  ndx1 = x coordinate of leftmost lattice points in this node */
    nx += 2;
/*  nx = number of lattice points in the x direction in one node */
    ny = nyt / nnd[1];
    if (coor[1] < nyt - ny * nnd[1])
/*      nyt is not a multiple of nnd[1], so add an extra row to this node */
    {   ny++;
        ndy1 = ny * coor[1] + 1;
    }
    else
        ndy1 = nyt - ny * (nnd[1] - coor[1]) + 1;
/*  ndy1 = y coordinate of bottom lattice points in this node */
    ndy2 = ndy1 + ny - 1;
/*  ndy2 = y coordinate of top lattice points in this node */
    ny += 2;
/*  ny = number of lattice points in the y direction in one node */
    n3 = nx - 1;
    n4 = -nx + 1;
    fp = fopen ("dout", "w");
    fprintf(fp, " coorx coory pnum  msknx mskny mskpx");
    fprintf(fp, " mskpy ndx1  ndy1  ndy2   nx    ny\n");
    fmulti (fp);
    fprintf(fp, "%6d%6d%6d%6d%6d%6d%6d%6d%6d%6d%6d\n", coor[0], coor[1],
    env.procnum, mskn[0], mskn[1], mskp[0], mskp[1], ndx1, ndy1, ndy2, nx, ny);
```

```
    fclose (fp);
    nd0[0] = 2 - (ndx1 + 1) % 3;
    nd0[1] = 2 - (ndy1 + 1) % 3;
    nd1[0] = 2 -  ndx1      % 3;
    nd1[1] = 2 -  ndy1      % 3;
    nd2[0] = 2 - (ndx1 + 2) % 3;
    nd2[1] = 2 - (ndy1 + 2) % 3;
    nde[0] = (ndx1 + 1) % 2;
    nde[1] = (ndy1 + 1) % 2;
    ndo[0] = ndx1 % 2;
    ndo[1] = ndy1 % 2;
    np = nx * ny;
/*  np = number of lattice points in one node */
    if (np > 4489)
/*      there are too many lattice points */
    {   printf("%6d = too many lattice points\n", np);
        exit(10);
    }
    npx = np - nx;
    npx2 = npx - nx;
    nx2 = nx * 2;
    nx3 = nx - 3;
    nxt2 = nxt * 2;
    nxt5 = nxt / 2;
    ny1 = ny - 1;
    ny2 = ny - 2;
    nb[0] = 4 * sizeof(long) + 2 * sizeof(short);
/*  nb[0] = number of bytes in a data item transferred in the x direction */
    nb[1] = nx * nb[0];
/*  nb[1] = number of bytes in a data item transferred in the y direction
          = offset in bytes between data items transferred in the x direction */
    lx.ni = nx2 - 1;
/*  lx.ni = lattice index for input of data moving in negative x direction */
    lx.no = nx + 1;
/*  lx.no = lattice index for output of data moving in negative x direction */
    lx.pi = nx;
```

```
/*  lx.pi = lattice index for input of data moving in positive x direction */
    lx.po = nx2 - 2;
/*  lx.po = lattice index for output of data moving in positive x direction */
    ly.ni = npx;
/*  ly.ni = lattice index for input of data moving in negative y direction */
    ly.no = nx;
/*  ly.no = lattice index for output of data moving in negative y direction */
    ly.pi = 0;
/*  ly.pi = lattice index for input of data moving in positive y direction */
    ly.po = npx2;
/*  ly.po = lattice index for output of data moving in positive y direction */
    dl3 = (double)idl * 4096.;
    dy1 = (double)idy * 16.;
    dy3 = (double)idy * 4096.;
    for (m0 = 0; m0 < np; m0++)
    {   la[m0].ipx = 0;
        la[m0].ipy = 0;
        la[m0].ivx = 0;
        la[m0].ivy = 0;
        la[m0].mlp = 0;
        la[m0].nlp = 0;
    }
    if (coor[1] == 0)
/*      this is a bottom row node, establish the bottom row wall */
    {   for (m0 = nx + 1; m0 < nx2 - 1; m0++)
        {   la[m0].ivx = jx1;
            la[m0].mlp = -1;
        }
    }
    if (coor[1] == nnd[1] - 1)
/*      this is a top row node, establish the top row wall */
    {   for (m0 = npx2 + 1; m0 < npx - 1; m0++)
        {   la[m0].ivx = jx2;
            la[m0].mlp = -1;
        }
    }
```

```
/*
    read in the initial data
*/
    if ((fp = fopen (nam2, "r")) == 0)
/*      fp = 0, the file cannot be opened */
    {   printf("Cannot open %s\n", nam2);
        exit(0);
    }
    fscanf (fp, "%*s%*s%6d\n", &nl);
    nm = 0;
/*  nm = number of marked movable particles */
    for (nn = 0; nn < nl; nn++)
    {   fscanf (fp, "%2d%6d%6d%6d%6d%7d%7d\n",
            &md, &ipx1, &ipy1, &ipx2, &ipy2, &ivxi, &ivyi);
/*      md = mode of particle(s)
          = -6 for a rectangular area of wall particles at a density of 1/3
          = -3 for a rectangular area of wall particles at a density of 1.0
          = -2 for a line of wall particles
          = -1 for a single wall particle
          =  0 for an empty position
          =  1 for a single movable particle
          =  2 for a line of movable particles
          =  3 for a rectangular area of movable particles at a density of 1.0
          =  6 for a rectangular area of movable particles at a density of 1/3
          > 10 for marked movable particles
        ipx1 = initial x position of particle, line, or area
        ipy1 = initial y position of particle, line, or area
        ipx2 = end x position of line or area or position offset of a particle
        ipy2 = end y position of line or area or position offset of a particle
        ivxi = initial x velocity of particle(s)
        ivyi = initial y velocity of particle(s)
*/
        if (md == 6 || md == -6)
/*          fill the area with particles at a density of 1/3 */
        {   md = (md > 0) ? 1 : -1;
            for (iy = ipy1; iy <= ipy2; iy++)
```

```
            {   if (iy < ndy1 || iy > ndy2) continue;
/*                  iy is outside this node, skip it */
                if ((iy - ipy1) % 2 == 1)
/*                  this is a displaced row */
                {   if (iy % 2 == 1)
/*                      this is an odd row */
                        iox = 2;
                    else
/*                      this is an even row */
                        iox = 1;
                }
                else
/*                  this is not a displaced row */
                    iox = 0;
                for (ix = ipx1 + iox; ix <= ipx2; ix += 3)
                {   m0 = (ix - ndx1 - (iy - 1) / 2 + nxt2) % nxt;
                    if (m0 > nx3) continue;
/*                      ix is outside this node, skip it */
                    m0 = m0 + 1 + (iy - ndy1 + 1) * nx;
                    la[m0].ivx = ivxi;
                    la[m0].ivy = ivyi;
                    la[m0].mlp = md;
                }
            }
        }
        if (md == 3 || md == -3)
/*          fill the area with particles at a density of 1.0 */
        {   md = (md > 0) ? 1 : -1;
            for (iy = ipy1; iy <= ipy2; iy++)
            {   if (iy < ndy1 || iy > ndy2) continue;
/*                  iy is outside this node, skip it */
                for (ix = ipx1; ix <= ipx2; ix++)
                {   m0 = (ix - ndx1 - (iy - 1) / 2 + nxt2) % nxt;
                    if (m0 > nx3) continue;
/*                      ix is outside this node, skip it */
                    m0 = m0 + 1 + (iy - ndy1 + 1) * nx;
```

```
                la[m0].ivx = ivxi;
                la[m0].ivy = ivyi;
                la[m0].mlp = md;
            }
        }
    }
    else if (md == 2 || md == -2)
/*          fill the line with particles */
    {   md = (md > 0) ? 1 : -1;
        idp[0] = ipx2 - ipx1;
        idp[1] = ipy2 - ipy1;
        ix = (idp[0] < 0) ? (-idp[0]) : idp[0];
        iy = (idp[1] < 0) ? (-idp[1]) : idp[1];
        npl = (ix > iy) ? ix : iy;
        for (ll = 0; ll <= npl; ll++)
        {   iy = ipy1 + idp[1] * ll / npl;
            if (iy < ndy1 || iy > ndy2) continue;
/*              iy is outside this node, skip it */
            ix = ipx1 + idp[0] * ll / npl;
            m0 = (ix - ndx1 - (iy - 1) / 2 + nxt2) % nxt;
            if (m0 > nx3) continue;
/*              ix is outside this node, skip it */
            m0 = m0 + 1 + (iy - ndy1 + 1) * nx;
            la[m0].ivx = ivxi;
            la[m0].ivy = ivyi;
            la[m0].mlp = md;
        }
    }
    else
/*          insert or remove a single particle */
    {   if (md > 10) nm++;
/*          this is a marker particle, number it accordingly */
        if (ipy1 >= ndy1 && ipy1 <= ndy2)
/*          the particle is in the same row as this node */
        {   m0 = (ipx1 - ndx1 - (ipy1 - 1) / 2 + nxt2) % nxt;
            if (m0 <= nx3)
```

```
/*                  the particle is in this node */
                {   m0 = m0 + 1 + (ipy1 - ndy1 + 1) * nx;
                    la[m0].ipx = ipx2;
                    la[m0].ipy = ipy2;
                    la[m0].ivx = ivxi;
                    la[m0].ivy = ivyi;
                    if (md > 10)
/*                      this is a marked, movable particle */
                    {   la[m0].nlp = nm;
                        la[m0].mlp = 1;
                    }
                    else
/*                      this is an unmarked particle */
                        la[m0].mlp = md;
                }
            }
        }
    }
    fclose (fp);
/*
    count the total number of movable grains ng,
    and count the number of extra wall particles nw,
    and find the peak velocities pv1, pv2, pvx, pvy. */
    ng = 0;
    nw = 0;
    pv.ipv2 = 0;
    pv.ipvx = 0;
    pv.ipvy = 0;
/*  ng = number of movable grains
    nw = number of extra wall grains
    pv.ipv2 = peak velocity**2
    pv.ipvx = x component of peak velocity
    pv.ipvy = y component of peak velocity
*/
    for (m0 = nx; m0 < npx; m0++)
    {   if (m0 % nx == 0 || (m0 + 1) % nx == 0) continue;
```

```
/*          this is the left or right column of the node, skip it */
        if (la[m0].mlp > 0)
/*          there is a movable particle at m0 */
        {   ng += 1;
            iv2p = la[m0].ivx * la[m0].ivx + la[m0].ivy * la[m0].ivy;
            if (iv2p > 0)
/*              there is a moving particle at m0 */
            {   if (iv2p > pv.ipv2)
/*                  this is a higher velocity */
                {   pv.ipvx = la[m0].ivx;
                    pv.ipvy = la[m0].ivy;
                    pv.ipv2 = iv2p;
                }
            }
        }
        else if (la[m0].mlp < 0)
/*          there is a wall particle at m0 */
            nw += 1;
    }
    combine (&ng, iadd, sizeof(long), 1);
    combine (&nw, iadd, sizeof(long), 1);
    combine (&pv, imax, 3*sizeof(long), 1);
    nw -= nxt2;
    rng = (double)ng;
    pv2 = (double)pv.ipv2;
    pv1 = sqrt(pv2);
/*
    create output files */
    if (ist > 1 && nm > 0)
/*      create the output file for marked movable particle data */
    {   fscanf (fi, "%s", nam5);
        fm = fopen (nam5, "w");
        fprintf(fm, "  nl =%6d\n", nm * nt + nw);
        fmulti (fm);
    }
    fclose(fi);
```

```
/*  create the output file for the step summary data */
    fp = fopen (nam3, "w");
/*  buffer the output file for the step summary data in blocks of 4096 bytes */
    setvbuf(fp, (char *)0, _IOFBF, 4096);
    fprintf(fp, "  nn   it    itt    ipv   srav2   avt   ");
    fprintf(fp, " avx    avy    is1    ip1    is2    ip2 \n");
/*
    start main time step iteration loop
*/
    for (nn = 1; nn <= nt; nn++)
/*
        calculate the time increment it */
    {   if (gm == 0.)
/*          calculate it based on the peak velocity pv1 */
            it = (pv1 > 0.) ? (int)(dl3 / pv1 + 0.5) : 1;
        else
/*          calculate it based on the peak velocity pv1, on the dot product
            of the peak velocity and the gravitational acceleration gv,
            and on the magnitude of the gravitational acceleration gm */
        {   gv = (double)(igx * pv.ipvx + igy * pv.ipvy);
/*          calculate the minimum value of it */
            it = (int)(2. * dy3 / (sqrt(pv2 + 2. * dy1 * gm) + pv1) + 0.5) - 1;
            rt = (double)(it) / 256.;
            dp = rt * sqrt((.25 * gm2 * rt + gv) * rt + pv2) - dy1;
            do
            {   dl = dp;
                it++;
                rt = (double)(it) / 256.;
                dp = rt * sqrt((.25 * gm2 * rt + gv) * rt + pv2) - dy1;
            }   while (dp < 0. && it < itm);
            if (dp > -dl) it--;
/*              the overshoot exceeds the undershoot, use the smaller it */
        }
        itt += it;
/*
        calculate the effects of gravity and update position offsets */
```

```
        gt[0] = (double)(igx * it) / 256.;
        igt[0] = (int)(gt[0] + ((gt[0] < 0) ? -0.5 : 0.5));
        idp[0] = (int)(gt[0] / 2. + ((gt[0] < 0) ? -0.5 : 0.5));
        gt[1] = (double)(igy * it) / 256.;
        igt[1] = (int)(gt[1] + ((gt[1] < 0) ? -0.5 : 0.5));
        idp[1] = (int)(gt[1] / 2. + ((gt[1] < 0) ? -0.5 : 0.5));
        for (m0 = nx; m0 < npx; m0++)
        {   if (m0 % nx == 0 || (m0 + 1) % nx == 0 || la[m0].mlp != 1) continue;
/*              this is the left or right column of the node, or
                this is not a movable particle, skip it */
            la[m0].ipx += ((la[m0].ivx + idp[0]) * it) / 4096;
            la[m0].ipy += ((la[m0].ivy + idp[1]) * it) / 4096;
            la[m0].ivx += igt[0];
            la[m0].ivy += igt[1];
        }
/*
        update the particle positions ntu times */
        for (nnp = 0; nnp < ntu; nnp++)
/*          do the position (0,0) template particles */
        {   shft();
            for (iy = nd0[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd0[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (2,1) template particles */
            shft();
            for (iy = nd1[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd2[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (1,2) template particles */
            shft();
            for (iy = nd2[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd1[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (2,0) template particles */
            shft();
            for (iy = nd0[1]; iy < ny; iy += 3)
```

```
            for (m0 = iy * nx + nd2[0]; m0 < iy * nx + nx; m0 += 3)
                if (la[m0].mlp == 1) uppo();
/*          do the position (1,1) template particles */
            shft();
            for (iy = nd1[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd1[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (0,2) template particles */
            shft();
            for (iy = nd2[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd0[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (1,0) template particles */
            shft();
            for (iy = nd0[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd1[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (0,1) template particles */
            shft();
            for (iy = nd1[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd0[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
/*          do the position (2,2) template particles */
            shft();
            for (iy = nd2[1]; iy < ny; iy += 3)
                for (m0 = iy * nx + nd2[0]; m0 < iy * nx + nx; m0 += 3)
                    if (la[m0].mlp == 1) uppo();
        }
/*
        calculate the effects of collisions, and update velocities */
        if (nn % 2 == 0)
/*          this is an even time step */
/*          do the position (0,0) template particles */
        {   shfb();
            for (iy = nde[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
```

```
            if (la[m0].mlp != 0) colb();
        shfc();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colc();
        shft();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colf();
/*      do the position (1,1) template particles */
        shft();
        for (iy = ndo[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colf();
        shfc();
        for (iy = ndo[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colc();
        shfb();
        for (iy = ndo[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colb();
/*      do the position (1,0) template particles */
        shfb();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colb();
        shfc();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colc();
        shft();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colf();
/*      do the position (0,1) template particles */
```

```
            shft();
            for (iy = ndo[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colf();
            shfc();
            for (iy = ndo[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colc();
            shfb();
            for (iy = ndo[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colb();
        }
        else
/*          this is an odd time step */
/*          do the position (0,1) template particles */
        {   shfb();
            for (iy = ndo[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colb();
            shfc();
            for (iy = ndo[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colc();
            shft();
            for (iy = ndo[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colf();
/*          do the position (1,0) template particles */
            shft();
            for (iy = nde[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                    if (la[m0].mlp != 0) colf();
            shfc();
            for (iy = nde[1]; iy < ny; iy += 2)
                for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
```

```
                if (la[m0].mlp != 0) colc();
        shfb();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colb();
/*      do the position (1,1) template particles */
        shfb();
        for (iy = ndo[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colb();
        shfc();
        for (iy = ndo[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colc();
        shft();
        for (iy = ndo[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + ndo[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colf();
/*      do the position (0,0) template particles */
        shft();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colf();
        shfc();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colc();
        shfb();
        for (iy = nde[1]; iy < ny; iy += 2)
            for (m0 = iy * nx + nde[0]; m0 < iy * nx + nx; m0 += 2)
                if (la[m0].mlp != 0) colb();
    }
/*
    calculate wall pressure tensor */
    is1 = 0;
    ip1 = 0;
```

```
        is2 = 0;
        ip2 = 0;
/*      is1 = x component of velocity changes absorbed by bottom wall particles
        ip1 = y component of velocity changes absorbed by bottom wall particles
        is2 = x component of velocity changes absorbed by top wall particles
        ip2 = y component of velocity changes absorbed by top wall particles
*/
        if (coor[1] == 0)
/*          this is a bottom wall node */
        {   for (m0 = nx + 1; m0 < nx2 - 1; m0++)
            {   is1 += la[m0].ipx;
                la[m0].ipx = 0;
                ip1 += la[m0].ipy;
                la[m0].ipy = 0;
            }
        }
        if (coor[1] == nnd[1] - 1)
/*          this is a top wall node */
        {   for (m0 = npx2 + 1; m0 < npx - 1; m0++)
            {   is2 += la[m0].ipx;
                la[m0].ipx = 0;
                ip2 += la[m0].ipy;
                la[m0].ipy = 0;
            }
        }
        combine (&is1, iadd, sizeof(long), 1);
        combine (&ip1, iadd, sizeof(long), 1);
        combine (&is2, iadd, sizeof(long), 1);
        combine (&ip2, iadd, sizeof(long), 1);
        is1 /= nxt;
        ip1 /= nxt;
        is2 /= nxt;
        ip2 /= nxt;
/*
        calculate the average velocities:
        av[0] = average velocity in the x direction,
```

```
        av[1] = average velocity in the y direction,
        av[2] = average velocity**2,
        av[3] = average thermal velocity,
        and calculate the forces is3 and ip3 on the extra wall particles,
        and count the number of particles in the right half nq,
        and find the peak velocities pv1, pv2, pvx, pvy. */
        is3 = 0;
        ip3 = 0;
        nq = 0;
/*      is3 = x component of velocity changes absorbed by extra wall particles
        ip3 = y component of velocity changes absorbed by extra wall particles
        nq = number of particles in the right half
*/
        pv.ipv2 = 0;
        pv.ipvx = 0;
        pv.ipvy = 0;
        vtt[0] = 0.;
        vtt[1] = 0.;
        vtt[2] = 0.;
/*      vtt[0] = total of x components of moving particle velocities
        vtt[1] = total of y components of moving particle velocities
        vtt[2] = total of square of moving particle velocities
*/
        shft();
        for (m0 = nx; m0 < npx; m0++)
        {   if (m0 % nx == 0 || (m0 + 1) % nx == 0) continue;
/*              this is the left or right column of the node, skip it */
            if ((ist > 1 && la[m0].nlp > 0) || (nw > 0 && la[m0].mlp > 0))
/*              calculate particle's coordinates */
            {   iy = m0 / nx;
                ix = m0 - iy * nx + ndx1 - 1;
                iy += ndy1 - 1;
                ix = (ix + (iy - 1) / 2 + nxt - 1) % nxt + 1;
            }
            if (ist > 1 && la[m0].nlp > 0)
/*              there is a marked movable particle at m0, write it to nam5 */
```

```
            fprintf(fm, "%2d %5d %5d %5d %5d %6d %6d %4d %3d\n",
                    la[m0].nlp + 10, ix, iy, la[m0].ipx, la[m0].ipy,
                    la[m0].ivx, la[m0].ivy, nn, env.procnum);
        if (la[m0].mlp > 0)
/*          there is a movable particle at m0 */
        {   if (nw > 0 && ix > nxt5) nq++;
/*              the particle is in the right half, count it */
            if (la[m0].mlp == 2) la[m0].mlp--;
/*              the particle was moved this time step,
                make it eligible to move in the next time step */
            iv2p = la[m0].ivx * la[m0].ivx + la[m0].ivy * la[m0].ivy;
            if (iv2p > 0)
/*              there is a moving particle at m0 */
            {   vtt[0] += (double)la[m0].ivx;
                vtt[1] += (double)la[m0].ivy;
                vtt[2] += (double)iv2p;
                if (iv2p > pv.ipv2)
/*                  this is a higher velocity */
                {   pv.ipvx = la[m0].ivx;
                    pv.ipvy = la[m0].ivy;
                    pv.ipv2 = iv2p;
                }
            }
        }
        else if (la[m0].mlp < 0)
/*          there is a wall particle at m0 */
        {   is3 += la[m0].ipx;
            la[m0].ipx = 0;
            ip3 += la[m0].ipy;
            la[m0].ipy = 0;
        }
    }
    if (nw > 0)
/*      there are extra wall particles */
    {   combine (&is3, iadd, sizeof(long), 1);
        combine (&ip3, iadd, sizeof(long), 1);
```

```
            combine (&nq, iadd, sizeof(long), 1);
            is3 /= nw;
            ip3 /= nw;
        }
        combine (&pv, imax, 3*sizeof(long), 1);
        combine (vtt, dadd, sizeof(double), 3);
        av[0] = vtt[0] / rng;
        av[1] = vtt[1] / rng;
        av[2] = vtt[2] / rng;
        av[3] = av[2] - av[0] * av[0] - av[1] * av[1];
        av[3] = (av[3] > 0.) ? sqrt(av[3]) : 0.;
        pv2 = (double)pv.ipv2;
        pv1 = sqrt(pv2);
/*
        output results */
/*      write the step summary data to nam3 and to screen */
        ia[0] = (int)(av[0] + ((av[0] < 0) ? -0.5 : 0.5));
        ia[1] = (int)(av[1] + ((av[1] < 0) ? -0.5 : 0.5));
        ia[2] = (int)(sqrt(av[2]) + 0.5);
        ia[3] = (int)(av[3] + 0.5);
        ipv = (int)(pv1 + 0.5);
        fprintf(fp, "%5d%5d%8d %5d %6d %6d %6d %6d %6d %6d %6d %6d\n",
        nn, it, itt, ipv, ia[2], ia[3], ia[0], ia[1], is1, ip1, is2, ip2);
        if (nw > 0)
/*          there are extra wall particles, output is3, ip3 and nq */
            fprintf(fp, " is3 = %6d  ip3 = %6d  nq = %7d\n", is3, ip3, nq);
        if (nn % nto == 0)
/*          write the particle data to nam4 */
        {   m1 = nx;
            m2 = npx;
            ny0 = (ndy1 - 2) / mvy;
/*          ny0 = y-index of lowest block of average velocities in this node */
            if (coor[1] == 0)
/*              this node is in the bottom row of nodes */
            {   m1 = nx2;
                ny0 = 0;
```

```
        }
        if (coor[1] == nnd[1] - 1) m2 = npx2;
/*          this node is in the top row of nodes */
        if (ist == 1 || ist == 3)
/*          average movable particle velocities over blocks */
        {   ng = 0;
/*          ng = count of lines in output file */
            for (ix = 0; ix <= (nxt - 1) / mvx; ix++)
            {   for (iy = ny0; iy <= (ndy2 - 2) / mvy; iy++)
                {   aivx[ix][iy] = 0;
                    aivy[ix][iy] = 0;
                    aivt[ix][iy] = 0.;
                    amlp[ix][iy] = 0;
                }
            }
            for (m0 = m1; m0 < m2; m0++)
            {   if (m0%nx==0 || (m0+1)%nx==0 || la[m0].mlp <= 0) continue;
/*                  this is the left or right column of the node, or
                    this is a wall particle, skip it */
                iy = m0 / nx;
                ix = m0 - iy * nx + ndx1 - 1;
                iy += ndy1 - 1;
                ix = (ix + (iy - 1) / 2 + nxt - 1) % nxt + 1;
                ix = (ix - 1) / mvx;
                iy = (iy - 2) / mvy;
                if (amlp[ix][iy] == 0) ng++;
/*                  this is now a non-zero particle block, count it */
                aivx[ix][iy] += la[m0].ivx;
                aivy[ix][iy] += la[m0].ivy;
                aivt[ix][iy] += (float)la[m0].ivx * (float)la[m0].ivx
                             +  (float)la[m0].ivy * (float)la[m0].ivy;
                amlp[ix][iy] ++;
            }
            combine (&ng, iadd, sizeof(long), 1);
        }
        if (ist < 4)
```

```
/*              output the particle data or the average block data */
            {   if (nto != nt) nam4[2] = nn / nto - 1 + 'a';
/*                  this is a multiple output run, rename the output file */
                fq = fopen (nam4, "w");
                fprintf(fq, "  nl =%6d\n", ng + nw);
                fmulti (fq);
            }
            for (m0 = m1; m0 < m2; m0++)
            {   if (m0%nx == 0 || (m0+1)%nx == 0 || la[m0].mlp == 0) continue;
/*                  this is the left or right column of the node, or
                    this lattice point is empty, skip it */
                if (ist != 0 && ist != 2 && la[m0].mlp > 0) continue;
/*                  this is not a movable particle output simulation, and
                    this lattice point is occupied by a movable particle,
                    skip it */
                iy = m0 / nx;
                ix = m0 - iy * nx + ndx1 - 1;
                iy += ndy1 - 1;
                ix = (ix + (iy - 1) / 2 + nxt - 1) % nxt + 1;
                if (ist < 4)
/*                  write particle data to nam4 */
                    fprintf(fq, "%2d %5d %5d %5d %5d %6d %6d %4d %3d\n",
                            la[m0].mlp, ix, iy, la[m0].ipx, la[m0].ipy,
                            la[m0].ivx, la[m0].ivy, m0, env.procnum);
                if (ist > 1 && nm > 0 && nn == nto && la[m0].mlp < 0)
/*                  write wall particle data to marker particle output file */
                    fprintf(fm, "%2d %5d %5d %5d %5d %6d %6d %4d %3d\n",
                            la[m0].mlp, ix, iy, la[m0].ipx, la[m0].ipy,
                            la[m0].ivx, la[m0].ivy, m0, env.procnum);
            }
            if (ist == 1 || ist == 3)
/*              print average movable particle position offsets
                and velocities over blocks in nam4 */
            {   nl = 0;
                for (ix = 0; ix <= (nxt - 1) / mvx; ix++)
                {   for (iy = ny0; iy <= (ndy2 - 2) / mvy; iy++)
```

```
                {   if (amlp[ix][iy] > 0)
/*                      there are any movable particles in this block,
                        print out the averages */
                    {   fprintf(fq, "-7 %5d %5d %8d %8d %11.5e %4d\n",
                                ix, iy, aivx[ix][iy], aivy[ix][iy],
                                aivt[ix][iy], amlp[ix][iy]);
                        nl++;
                    }
                }
            }
            combine (&nl, iadd, sizeof(int), 1);
            fsingl (fq);
            fprintf(fq, "  nl =%6d\n", nl);
        }
        if (ist < 4) fclose (fq);
/*          close the output file nam4 */
    }
    if (ist > 1 && nm > 0 && nn % 4 == 0) fflush (fm);
/*      there are marked particles, flush the marked particle output file */
  }
/*  end of main time step loop */
  ng = (int) rng;
  fprintf(fp, " ng = %7d  nw = %7d\n", ng, nw);
  fclose (fp);
  if (ist > 1 && nm > 0) fclose (fm);
/*     close the marker particle file */
  exit(0);
}
/*
   This function adds two doubles for the combine function
*/
dadd (a, b, size)
double *a, *b;
int size;
{   *a += *b;
    return 1;
```

```
}
/*
    This function adds two integers for the combine function
*/
iadd (i, j, size)
long *i, *j;
int size;
{   *i += *j;
    return 1;
}
/*
    This function finds the peak velocity**2 and
    its x and y components for the combine function
*/
imax (i, j, size)
struct pkvlct *i;
struct pkvlct *j;
int size;
{   if (i->ipv2 < j->ipv2)
/*      the peak velocity at j is higher, keep it */
    {   i->ipv2 = j->ipv2;
        i->ipvx = j->ipvx;
        i->ipvy = j->ipvy;
    }
    return 1;
}
/*
    This function shifts data between the nodes in the y direction
*/
shfb()
{   cshift (&la[ly.ni], mskp[1], nb[1],
            &la[ly.no], mskn[1], nb[1]);
    cshift (&la[ly.pi], mskn[1], nb[1],
            &la[ly.po], mskp[1], nb[1]);
}
/*
```

```
    This function shifts data between the nodes in the x direction
*/
shfc()
{   vshift (&la[lx.ni], mskp[0], nb[0], nb[1], ny2,
            &la[lx.no], mskn[0], nb[0], nb[1], ny2);
    vshift (&la[lx.pi], mskn[0], nb[0], nb[1], ny2,
            &la[lx.po], mskp[0], nb[0], nb[1], ny2);
}
/*
    This function shifts data between the nodes in all four directions
*/
shft()
{   vshift (&la[lx.ni], mskp[0], nb[0], nb[1], ny2,
            &la[lx.no], mskn[0], nb[0], nb[1], ny2);
    vshift (&la[lx.pi], mskn[0], nb[0], nb[1], ny2,
            &la[lx.po], mskp[0], nb[0], nb[1], ny2);
    cshift (&la[ly.ni], mskp[1], nb[1],
            &la[ly.no], mskn[1], nb[1]);
    cshift (&la[ly.pi], mskn[1], nb[1],
            &la[ly.po], mskp[1], nb[1]);
}
/*
    This function updates the position of the particle at m0
*/
uppo()
{   int icx, icy, islope, ix, iy;
    ix = la[m0].ipx;
    iy = la[m0].ipy;
    if (ix < idl && ix > -idl &&
        iy < idl && iy > -idl &&
        ix * ix + iy * iy < id)
/*      particle will not jump to a new position */
        la[m0].mlp++;
    else
/*      particle may jump to a new position */
    {   if (ix > 0)
```

```
/*          particle is moving in the +x direction */
        {   islope = 256 * iy / ix;
            if (islope <= -148)
/*              particle is moving towards 300 degrees */
            {   m1 =  n4;
                icx = -idx;
                icy =  idy;
            }
            else if (islope < 148)
/*              particle is moving towards 000 degrees */
            {   m1 =  1;
                icx = -id2;
                icy =    0;
            }
            else
/*              particle is moving towards 060 degrees */
            {   m1 =  nx;
                icx = -idx;
                icy = -idy;
            }
        }
        else if (ix < 0)
/*          particle is moving in the -x direction */
        {   islope = 256 * iy / ix;
            if (islope >= 148)
/*              particle is moving towards 240 degrees */
            {   m1 = -nx;
                icx =  idx;
                icy =  idy;
            }
            else if (islope > -148)
/*              particle is moving towards 180 degrees */
            {   m1 = -1;
                icx =  id2;
                icy =    0;
            }
```

```
            else
/*              particle is moving towards 120 degrees */
            {   m1 =  n3;
                icx =  idx;
                icy = -idy;
            }
        }
        else if (iy < 0)
/*          particle is moving in -y direction */
        {   if ((m0 / nx + ndy1) % 2 == 0)
/*              particle is on an odd row, move to position 6 */
            {   m1 =  n4;
                icx = -idx;
            }
            else
/*              particle is on an even row, move to position 5 */
            {   m1 = -nx;
                icx =  idx;
            }
            icy =  idy;
        }
        else
/*          particle is moving in +y direction */
        {   if ((m0 / nx + ndy1) % 2 == 0)
/*              particle is on an odd row, move to position 2 */
            {   m1 =  nx;
                icx = -idx;
            }
            else
/*              particle is on an even row, move to position 1 */
            {   m1 =  n3;
                icx =  idx;
            }
            icy = -idy;
        }
/*      eliminate particles in adjacent nodes which are not incoming */
```

```
        if ( m0    % nx ==  0  && m1 !=  1  && m1 != n4) return;
/*          this is  left column of node and particle is not going to d or f */
        if ((m0+1) % nx ==  0  && m1 != -1  && m1 != n3) return;
/*          this is right column of node and particle is not going to c or a */
        if ( m0    / nx ==  0  && m1 !=  nx && m1 != n3) return;
/*          this is bottom  row  of node and particle is not going to b or a */
        if ( m0    / nx == ny1 && m1 != -nx && m1 != n4) return;
/*          this is   top   row  of node and particle is not going to e or f */
        m1 += m0;
        if (la[m1].mlp == 0)
/*          space is empty, particle can move */
        {   la[m1].ipx = la[m0].ipx + icx;
            la[m0].ipx = 0;
            la[m1].ipy = la[m0].ipy + icy;
            la[m0].ipy = 0;
            la[m1].ivx = la[m0].ivx;
            la[m0].ivx = 0;
            la[m1].ivy = la[m0].ivy;
            la[m0].ivy = 0;
            la[m1].mlp = la[m0].mlp + 1;
            la[m0].mlp = 0;
            la[m1].nlp = la[m0].nlp;
            la[m0].nlp = 0;
        }
    }
}
/*
    This function calculates the effects of a collision between the
    particle at [m0] and any adjacent particle in the b direction
*/
colb()
{   long ikc, ikcx, ikcy, ip, kc1;
/*      ikc  = change of particle velocities in a collision * ke
        ikcx = x-component of ikc
        ikcy = y-component of ikc
        ip   = change in position offsets in a collision
```

```
        kc1  = relative particle velocity before collision
*/
    m1 = m0 + nx;
    if (m1 < np && m1 % nx != 0 && (m1 + 1) % nx != 0)
/*      do the collision with position b */
    {   if (la[m0].mlp < 0)
/*          the particle at [m0] is fixed */
        {   if (la[m1].mlp > 0)
/*              there is a movable particle at [m1], calculate k.c12 */
            {   kc1 = (kx * (la[m0].ivx - la[m1].ivx) +
                       ky * (la[m0].ivy - la[m1].ivy) + kx) / kq;
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iew1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
                    {   ikc = icw + iew2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
                        ikc += ke * kc1;
                    }
                    ikcx = (kx * ikc + kxw) / kqw;
                    ikcy = (ky * ikc + kxw) / kqw;
                    la[m0].ipx -= ikcx;
                    la[m1].ivx += ikcx;
                    la[m0].ipy -= ikcy;
                    la[m1].ivy += ikcy;
                    ip = (idy * la[m1].ipx - idx * la[m1].ipy) / id2;
                    la[m1].ipx =  ip * idy / id2;
                    la[m1].ipy = -ip * idx / id2;
                }
            }
        }
        else
/*          the particle at [m0] is movable */
```

```
        {   if (la[m1].mlp < 0)
/*              there is a fixed particle at [m1], calculate k.c12 */
            {   kc1 = (kx * (la[m0].ivx - la[m1].ivx) +
                       ky * (la[m0].ivy - la[m1].ivy) + kx) / kq;
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iew1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
                    {   ikc = icw + iew2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
                        ikc += ke * kc1;
                    }
                    ikcx = (kx * ikc + kxw) / kqw;
                    ikcy = (ky * ikc + kxw) / kqw;
                    la[m0].ivx -= ikcx;
                    la[m1].ipx += ikcx;
                    la[m0].ivy -= ikcy;
                    la[m1].ipy += ikcy;
                    ip = (idy * la[m0].ipx - idx * la[m0].ipy) / id2;
                    la[m0].ipx =  ip * idy / id2;
                    la[m0].ipy = -ip * idx / id2;
                }
            }
            else if (la[m1].mlp > 0)
/*              there is a movable particle at [m1], calculate k.c12 */
            {   kc1 = (kx * (la[m0].ivx - la[m1].ivx) +
                       ky * (la[m0].ivy - la[m1].ivy) + kx) / kq;
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iep1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
```

```
                    {   ikc = icp + iep2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
                        ikc += ke * kc1;
                    }
                    ikcx = (kx * ikc + kxp) / kqp;
                    ikcy = (ky * ikc + kxp) / kqp;
                    la[m0].ivx -= ikcx;
                    la[m1].ivx += ikcx;
                    la[m0].ivy -= ikcy;
                    la[m1].ivy += ikcy;
                    ip = (idy * la[m0].ipx - idx * la[m0].ipy) / id2;
                    la[m0].ipx =  ip * idy / id2;
                    la[m0].ipy = -ip * idx / id2;
                    ip = (idy * la[m1].ipx - idx * la[m1].ipy) / id2;
                    la[m1].ipx =  ip * idy / id2;
                    la[m1].ipy = -ip * idx / id2;
                }
            }
        }
    }
}
/*
    This function calculates the effects of a collision between the
    particle at [m0] and any adjacent particle in the c direction
*/
colc()
{   long ikc, ikcx, kc1;
/*      ikc  = change of particle velocities in a collision * ke
        ikcx = x-component of ikc
        kc1  = relative particle velocity before collision
*/
    m1 = m0 - 1;
    if (m0 > nx && m0 < npx && m0 % nx != 0)
/*      do the collision with position c */
    {   if (la[m0].mlp < 0)
```

```
/*          the particle at [m0] is fixed */
        {   if (la[m1].mlp > 0)
/*              there is a movable particle at [m1], calculate k.c12 */
            {   kc1 = la[m1].ivx - la[m0].ivx;
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iew1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
                    {   ikc = icw + iew2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
                        ikc += ke * kc1;
                    }
                    ikcx = (ikc + ke2) / ke;
                    la[m0].ipx += ikcx;
                    la[m1].ivx -= ikcx;
                    la[m1].ipx = 0;
                }
            }
        }
        else
/*          the particle at [m0] is movable */
        {   if (la[m1].mlp < 0)
/*              there is a fixed particle at [m1], calculate k.c12 */
            {   kc1 = la[m1].ivx - la[m0].ivx;
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iew1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
                    {   ikc = icw + iew2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
```

```
                    ikc += ke * kc1;
                }
                ikcx = (ikc + ke2) / ke;
                la[m0].ivx += ikcx;
                la[m1].ipx -= ikcx;
                la[m0].ipx = 0;
            }
        }
        else if (la[m1].mlp > 0)
/*          there is a movable particle at [m1], calculate k.c12 */
        {   kc1 = la[m1].ivx - la[m0].ivx;
            if (kc1 > 0)
/*              there will be a collision */
            {   if (kc1 <= icb) ikc = iep1 * kc1;
/*                  the collision velocity is below icb */
                else
/*                  the collision velocity is above icb */
                {   ikc = icp + iep2 * kc1;
                    if (ikc < 0) ikc = 0;
/*                      relative rebound velocity cannot be less than 0 */
                    ikc += ke * kc1;
                }
                ikcx = (ikc + ke) / ke1;
                la[m0].ivx += ikcx;
                la[m1].ivx -= ikcx;
                la[m0].ipx = 0;
                la[m1].ipx = 0;
            }
        }
    }
}
/*
    This function calculates the effects of a collision between the
    particle at [m0] and any adjacent particle in the f direction
*/
```

```
colf()
{   long ikc, ikcx, ikcy, ip, kc1;
/*      ikc  = change of particle velocities in a collision * ke
        ikcx = x-component of ikc
        ikcy = y-component of ikc
        ip   = change in position offsets in a collision
        kc1  = relative particle velocity before collision
*/
    m1 = m0 + n4;
    if (m1 > 0 && m1 % nx != 0)
/*      do the collision with position f */
    {   if (la[m0].mlp < 0)
/*          the particle at [m0] is fixed */
        {   if (la[m1].mlp > 0)
/*              there is a movable particle at [m1], calculate k.c12 */
            {   kc1 = (kx * (la[m0].ivx - la[m1].ivx) +
                       ky * (la[m1].ivy - la[m0].ivy) + kx) / kq;
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iew1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
                    {   ikc = icw + iew2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
                        ikc += ke * kc1;
                    }
                    ikcx = (kx * ikc + kxw) / kqw;
                    ikcy = (ky * ikc + kxw) / kqw;
                    la[m0].ipx -= ikcx;
                    la[m1].ivx += ikcx;
                    la[m0].ipy += ikcy;
                    la[m1].ivy -= ikcy;
                    ip = (idy * la[m1].ipx + idx * la[m1].ipy) / id2;
                    la[m1].ipx =  ip * idy / id2;
```

```
                la[m1].ipy =  ip * idx / id2;
            }
        }
    }
    else
/*      the particle at [m0] is movable */
    {   if (la[m1].mlp < 0)
/*          there is a fixed particle at [m1], calculate k.c12 */
        {   kc1 = (kx * (la[m0].ivx - la[m1].ivx) +
                   ky * (la[m1].ivy - la[m0].ivy) + kx) / kq;
            if (kc1 > 0)
/*              there will be a collision */
            {   if (kc1 <= icb) ikc = iew1 * kc1;
/*                  the collision velocity is below icb */
                else
/*                  the collision velocity is above icb */
                {   ikc = icw + iew2 * kc1;
                    if (ikc < 0) ikc = 0;
/*                      relative rebound velocity cannot be less than 0 */
                    ikc += ke * kc1;
                }
                ikcx = (kx * ikc + kxw) / kqw;
                ikcy = (ky * ikc + kxw) / kqw;
                la[m0].ivx -= ikcx;
                la[m1].ipx += ikcx;
                la[m0].ivy += ikcy;
                la[m1].ipy -= ikcy;
                ip = (idy * la[m0].ipx + idx * la[m0].ipy) / id2;
                la[m0].ipx =  ip * idy / id2;
                la[m0].ipy =  ip * idx / id2;
            }
        }
        else if (la[m1].mlp > 0)
/*          there is a movable particle at [m1], calculate k.c12 */
        {   kc1 = (kx * (la[m0].ivx - la[m1].ivx) +
                   ky * (la[m1].ivy - la[m0].ivy) + kx) / kq;
```

```
                if (kc1 > 0)
/*                  there will be a collision */
                {   if (kc1 <= icb) ikc = iep1 * kc1;
/*                      the collision velocity is below icb */
                    else
/*                      the collision velocity is above icb */
                    {   ikc = icp + iep2 * kc1;
                        if (ikc < 0) ikc = 0;
/*                          relative rebound velocity cannot be less than 0 */
                        ikc += ke * kc1;
                    }
                    ikcx = (kx * ikc + kxp) / kqp;
                    ikcy = (ky * ikc + kxp) / kqp;
                    la[m0].ivx -= ikcx;
                    la[m1].ivx += ikcx;
                    la[m0].ivy += ikcy;
                    la[m1].ivy -= ikcy;
                    ip = (idy * la[m0].ipx + idx * la[m0].ipy) / id2;
                    la[m0].ipx =  ip * idy / id2;
                    la[m0].ipy =  ip * idx / id2;
                    ip = (idy * la[m1].ipx + idx * la[m1].ipy) / id2;
                    la[m1].ipx =  ip * idy / id2;
                    la[m1].ipy =  ip * idx / id2;
                }
            }
        }
    }
}
```

## References

Alder, B.J., and Wainwright, T.E., 1959, "Studies in Molecular Dynamics. I. General Method," *The Journal of Chemical Physics*, **31** (2) 459–466.

Bagnold, R.A., 1954, "Experiments on a Gravity-Free Dispersion of Large Solid Spheres in a Newtonian Fluid under Shear," *Proceedings of the Royal Society of London*, Series A, **225** 49–63.

Campbell, C.S., and Gong, A., 1987, "Boundary Conditions for Two-Dimensional Granular Flows," *Proceedings of the International Symposium on Multiphase Flows*, Hangzhou, China, Volume 1, 278.

Chapman, S., and Cowling, T.G., 1970, *The Mathematical Theory of Non-Uniform Gases*, third edition, Cambridge University Press, Cambridge.

Cundall, P.A., and Strack, O.D.L., 1979, "A Discrete Numerical Model for Granular Assemblies," *Geotechnique*, **29** (1) 47–65.

Frisch, U., Hasslacher, B., and Pomeau, Y., 1986, "Lattice-Gas Automata for the Navier-Stokes Equation," *Physical Review Letters*, **56** (14) 1505–1508.

Goldstein, D., Sturtevant, B., and Broadwell, J.E., 1989, "Investigations of the Motion of Discrete-Velocity Gases," *Rarefied Gasdynamics*, ed. E.P. Muntz, AIAA.

Goodman, M.A., and Cowin, S.C., 1972, "A Continuum Theory for Granular Materials," *Archive for Rational Mechanics and Analysis*, **44** (4) 249–266.

de Groot, S.R., and Mazur, P., 1962, *Nonequilibrium Thermodynamics*, Dover Publications, New York.

Haff, P.K., 1983, "Grain Flow as a Fluid-Mechanical Phenomena," *Journal of Fluid Mechanics*, **134** 401–430.

Haff, P.K., 1987, "Micromechanical Aspects of Sound Waves in Granular Materials," *Proceedings of Solids Transport Contractor's Review Meeting*, (September 17–18, 1987) Pittsburgh, DOE, 41–67.

Hanes, D.M., 1987, "Boundary Grain Size Influence upon the Dynamics of Rapidly Flowing Glass Spheres," Sixth Engineering Mechanics Specialty Conference, Buffalo, New York.

Henderson, D., Abraham, F.F., and Barker, J.A., 1976, "The Ornstein-Zernike Equation for a Fluid in Contact with a Surface," *Molecular Physics*, **31** (4) 1291–1295.

Hirschfelder, J.O., Curtiss, D.F., and Bird, R.F., 1964, *Molecular Theory of Gases and Liquids*, Wiley.

Hui, K., Haff, P.K., Ungar, J.E., and Jackson, R., 1984, "Boundary Conditions for High Shear Rate Grain Flows," *Journal of Fluid Mechanics*, **145** 223–233.

Jackson, R., 1986, "Some Features of the Flow of Granular Materials and Aerated Granular Materials," *Journal of Rheology*, **30** 907–930.

Jenkins, J.T., and Richman, M.W., 1986, "Boundary Conditions for Plane Flows of Smooth, Nearly Elastic, Circular Disks," *Journal of Fluid Mechanics*, **171** 53–69.

Jenkins, J.T., and Savage, S.B., 1983, "A Theory for the Rapid Flow of Identical, Smooth, Nearly Elastic, Spherical Particles," *Journal of Fluid Mechanics*, **130** 187–202.

Johnson, P.C., and Jackson, R., 1987, "Frictional-Collisional Constitutive Relations for Granular Materials, with Application to Plane Shearing," *Journal of Fluid Mechanics*, **176** 67–93.

Kanatani, K., 1979, "A Micropolar Continuum Theory for the Flow of Granular Materials," *International Journal of Engineering Science*, **17** 419–432.

Lun, C.K.K., Savage, S.B., Jeffery, D.J., and Chepurniy, N., 1984, "Kinetic Theories for Granular Flow: Inelastic Particles in Couette Flow and Slightly Inelastic Particles in a General Flow Field," *Journal of Fluid Mechanics*, **140** 223–256.

Lun, C.K.K., and Savage, S.B., 1987, "A Simple Kinetic Theory for Granular Flow of Rough, Inelastic, Spherical Particles," *Journal of Applied Mechanics*, **54** 47–53.

Margolis, N., Tommaso, T., and Vichniac, G., 1986, "Cellular-Automata Supercomputers for Fluid-Dynamics Modeling," *Physical Review Letters*, **56** (16) 1694–1696.

Nedderman, R.M., Davies, S.T., and Horton, D.J., 1980, "The Flow of Granular Materials Around Obstacles," *Powder Technology*, **25** 215–223.

Nunziato, J.W., Passman, S.L., and Thomas, J.P., Jr., 1980, "Gravitational Flows of Granular Materials with Incompressible Grains," *Journal of Rheology*, **24** (4) 395–420.

Ogawa, S., Umemura, A., and Oshima, N., 1980, "On the Equations of Fully Fluidized Granular Materials," *Z. angew. Math. Phys.*, **31** 483–493.

Oshima, N., 1978, "Dynamics of Fluidized Granular Media," in *Theoretical and Applied Mechanics*, **28** 475–484.

Passman, S.L., Nunziato, J.W., Bailey, P.B., and Thomas, J.P., Jr., 1980, "Shearing Flows of Granular Materials," *Journal of the Engineering Mechanics Division, ASCE*, **106** 773–783.

Savage, S.B., 1979, "Gravity Flow of Cohesionless Granular Materials in Chutes and Channels," *Journal of Fluid Mechanics*, **92** 53–96.

Savage, S.B., and Jeffrey, D.J., 1981, "The Stress Tensor in a Granular Flow at High Shear Rates," *Journal of Fluid Mechanics*, **110** 255–272.

Shen, H.H., and Ackermann, N.L., 1984, "Constitutive Equations for a Simple Shear Flow of a Disk Shaped Granular Mixture," *International Journal of Engineering Science*, **22** 829–843.

Snook, I.K., and Henderson, D., 1978, "Monte Carlo Study of a Hard-Sphere Fluid Near a Hard Wall," *Journal of Chemical Physics*, **68** (5) 2134–2139.

Tuzun, U., Nedderman, R.M., 1982, "An Investigation of the Flow Boundary During Steady-State Discharge from a Funnel-Flow Bunker," *Powder Technology*, **31** 27–43.

Vichniac, G.Y., 1984, "Simulating Physics with Cellular Automata," *Physica*, **10D** 96–116.

Waisman, E., Henderson, D., and Lebowitz, J.L., 1976, "Solution of the Mean Spherical Approximation for the Density Profile of a Hard Sphere Fluid Near a Hard Wall," *Molecular Physics*, **32** (5) 1373–1381.

Walton, O.R., 1984, "Computer Simulation of Particulate Flow," *Energy and Technology Review (Lawrence Livermore National Laboratory)*, (May) 24–36.

Werner, B.T., 1987, "A Physical Model of Wind-Blown Sand Transport," *Caltech Basic and Applied Physics: Brown Bag Thesis*, (April) 442p.