PREDICTION OF SCANNING TUNNELING MICROSCOPE IMAGES BY COMPUTATIONAL QUANTUM CHEMISTRY: CHEMICAL MODELS AND SOFTWARE DESIGN

Thesis by

Terry Ronald Coley

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1993

(Submitted May 10, 1993)

Acknowledgments

More than anyone else professor William A. Goddard, III is responsible for creating the remarkable research environment in which I have had the privilege to work. The amount of stimulating exposure to leading edge computing tools enjoyed by the members of his group is hard to equal. I am grateful for his support in this respect. I am also grateful for the independence he has fostered in me.

Professor John D. Baldeschwieler has impressed me with, among other things, his willingness to invest in scientific endeavors with a long term payoff. Surely, this project fits that category. I am grateful for his continued support.

Many people in the Goddard and Baldeschwieler groups as well as others at Caltech have contributed to the enjoyment and learning in my experience at Caltech, especially, Mark Brusich, Emily Carter, Steve Cuccaro, Siddharth Dasgupta, Robert Driscoll, Jonathan Hurley, John Kramar, Jean-Marc Langlois, Kian-Tat Lim, Alan Mathiowetz, Rick Muller, Murco Ringnalda, Robert Scheid, Mike Weimer, Mike Youngquist, and Dan Zirin.

For funding, I gratefully acknowledge contributions from the following: NSF, NIH, AFOSR, Hughes Aircraft Co.

My fiancee, Laurie Carlson, has been very encouraging. Thanks, Laurie.

Finally, I thank my parents for making this all possible.

Abstract

We have created chemical models for predicting and interpreting STM images of several specific systems. Detailed studies are made of transition metal dichalcogenides (MoS_2 and $MoTe_2$), Xe on Ni (110), C_3H_4 on Ni (110), and *n*-butyl benzene on a graphite model ($C_{42}O_6H_{12}$). In the case of MoS_2 we study the ambiguity in the STM images regarding the assignment of peaks to the subsurface metal or the surface chalcogenide. In the Ni models we study STM imaging mechanisms for cases where the adsorbate states lie far above and below the metal Fermi level. The large *n*-butyl on graphite system models a system where adsorbate states can play a direct role in the imaging. Results from the cluster studies are related to various STM imaging modes, including constant current mode, constant height mode, and barrier height imaging.

Two new procedures are developed to aid in computational prediction of STM images. First, we implement an algorithm for computing Bardeen-type tunneling matrix elements from *ab initio* wave functions in Gaussian basis sets. Second, we show how to obtain state densities as a function of energy for bulk substrate/adsorbate systems using only Fock matrix elements from cluster calculations. Initial results are presented for a linear chain of Ni atoms with a perturbing Xe atom.

A software environment for computational chemistry developed in the course of performing these calculations is presented. Tools for creating computational servers to perform chemistry calculations are described. Embedded in each chemistry server is a public domain control language created by J. Ousterhout at the University of California, Berkeley. This allows the development of a variety of clients for controlling the servers using a common language. Clients can be simple text "scripts" that organize a calculation,

graphical interfaces, or control streams from other programs. All software entities are designed in an object oriented fashion discussed in the text.

The tools provide two significant advantages to computational chemists over traditional approaches. First, we show how to prototype new algorithms with little or no compiled code thus speeding up the debugging process. Second, the client-server design of the software components makes it natural to create distributed parallel applications.

Table of Contents

1. Introduction to Thesis

There are two foci of this thesis. First, we develop methods and guidelines for creating computationally tractable chemical models to predict scanning tunneling microscope (STM) images. Second, we design and implement an entirely new software environment for computational chemistry that is shown through case studies to enhance productivity in scientific programming. This introduction discusses STM models; discussion of software design comes later.

We hope this thesis presents an approachable picture to the STM experimentalist. Two of our long term goals have been to develop the rules of thumb and improve the computational tools to a state where experimentalists and theoreticians can use their chemical intuition to ask and answer questions about STM imaging. We also intend this work to present some alternative ways of thinking for the developer of research computational chemistry programs. Every effort has been made to present these alternatives with examples which demonstrate an enhancement of the creative freedom and productivity of the scientist/programmer.

1.1. STM Background

Although good approximate theories exist for understanding current flow in the STM, these have been remarkably devoid of chemically specific applications. The lack of chemically specific theoretical tools to complement STM experimental results has hampered image interpretation in some cases. It also limits our ability to predict what modifications to an STM sample or tip would lead to improved images and to images that tell us about specific chemical features of the sample (e.g., labeling). To address these issues we have developed models for particular STM sample and sample/adsorbate systems and have used *ab initio* molecular quantum mechanics programs to generate

electronic wave functions with enough detail to discern the chemically important aspects of STM images.

The highly localized variations of the STM tunneling current result in atomic resolution for many systems. *Ab initio* cluster models describe the electronic wave function in a localized area particularly well (sufficient for a vast literature of chemically accurate predictions). Hartree-Fock and Generalized Valence Bond (GVB) computational techniques are a natural choice for describing some aspects of the STM imaging mechanism, especially for systems involving adsorbate states and local barrier height images.

Experience gained from our work on cluster models for STM systems has led to a few simple guidelines for creating good STM models. First, one must create a cluster with sufficient size to adequately describe the chemical environment of the tip-sample junction. Second, scaled out (more flexible) basis sets are sometimes needed to properly describe the exponentially decaying vacuum tails of wave functions. Achieving a proper chemical environment is reasonably intuitive to the chemist looking at the problem. At the very least, it means including all nearest neighbors around the atom or atoms of interest. Achieving proper exponential decay is a technical point: standard basis sets are optimized for bonding situations. The STM tip-sample distance exceeds these distances, and the use of Gaussian basis sets, $\exp(-\alpha r^2)$, leads to improperly rapid die-off in the vacuum. This must be checked for in the calculations.

While much useful information can be gained from a normal (though usually large) Hartree-Fock or GVB calculation, more information would be useful. For example, interpreting STM images from the calculated molecular orbitals relies on the adequacy of the simplest model of the STM tunneling current (Tersoff and Hamann,[52] described below) in which the current is proportional to the density of contributing sample states at

the position of the tip. Also, relatively small cluster models divulge no information on the density of states as a function of energy for the bulk substrate (or tip). As we will see, the density of states term contributes to the STM tunneling current.

To address these issues we have taken three approaches for which results are presented. First, we have calculated tunneling matrix elements using a method developed by Voter and Goddard[8] for obtaining the Hamiltonian matrix element between two many-electron Hartree-Fock or GVB wave functions. Second, we have developed a new computational procedure for calculating Bardeen[51] type tunneling matrix elements. The advantages and disadvantages of these two approaches to tunneling matrix element computations are discussed. Finally, we have begun the development of a method to obtain densities of states as a function of energy for sample/adsorbate systems using only information from *ab initio* clusters. This completes the picture for *ab initio* STM image prediction. Initial results are presented.

The remainder of this introduction provides a few basic principles needed to understand what follows. For a brief introduction to the modes of STM operation we will address theoretically, see the thesis Appendix.

1.1.1. One-Dimensional Tunneling Model

For a one-dimensional square barrier of height ϕ above the Fermi level and a small bias, V (shown in the schematic below), the tunneling current, I, is proportional to

$$I \propto Ve^{-1.025s\sqrt{\phi}} \tag{1.1}$$

where the barrier height is in eV and the barrier thickness, s, is in Å[1] In atomic units, the same proportionality is given by $I \propto Ve^{-2s\sqrt{2\phi}}$.

---

[1]Lang, N. D., *Phys. Rev. B*, **37** 10395 (1988).

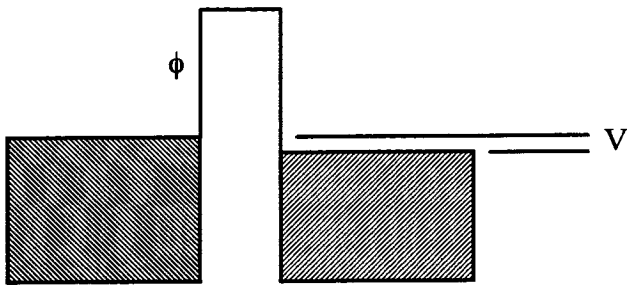


Figure 1:
Electrons tunnel from occupied states on the left to unoccupied states on the right.

Using Eq. (1.1) the barrier height may be expressed as

$$\phi(eV) = 0.952\left(\frac{d\ln I}{ds(\text{Å})}\right)^2 \tag{1.2}$$

or in atomic units

$$\phi = \frac{1}{8}\left(\frac{d\ln I}{ds}\right)^2.$$

Of course, in real STM experiments the barrier is not exactly square; nevertheless, Eq. (1.2) allows the definition of an *effective* barrier height and is often used in the experimental literature.

From Eq. (1.1) we can make the following simple analysis of the tunneling current: topographic information arises due to the exponential dependence on s, the tip-sample separation. Chemical information is contained in the ϕ dependence. State density information arises in the pre-exponential factor.[2] To quantitatively predict the tunneling current, we must address each of these terms together. However, taken separately we can still obtain useful chemical and structural information without absolute currents if some assumptions are made about the unknown parameters.

---

[2]J. M. Gómez-Rodríguez, J. Gómez-Herrero, and A. M. Baró, *Surf. Sci.*, **220**, 152-164, (1989).

1.1.2. Average Local Ionization Potentials

Define the average local ionization potential of a Hartree-Fock wave function by

$$\bar{I}(r) = \sum_{a}^{occupiedMOs} \frac{-\varepsilon_a \rho_a(r)}{\rho(r)} = \sum_{a}^{occupiedMOs} \frac{-\varepsilon_a \varphi_a^2(r)}{\sum_{b}^{occupiedMOs} \varphi_b^2(r)} \tag{1.3}$$

where ε_a is the Koopmans' ionization potential[3] of orbital *a*. Since $\rho(r)$ is the total electron density at the point *r*, Eq. (1.3) defines $\bar{I}(r)$ as a (probability) density weighted average of the ionization potential at the point *r*. That is, $\bar{I}(r)$ is the average energy required to remove an electron from a point in space near the molecule. In the language of STM this might be called an average local barrier height.

Sjoberg *et al.*,[4] have shown a correlation between $\bar{I}(r)$ and preferred sites for electrophilic substitution in aromatics (substituted benzenes). Similarly, Brinck *et al.*,[5] have demonstrated a correlation between $\bar{I}(r)$ and pK_a values in a variety of azines and azoles.

1.1.3. Average Local Tunneling Barriers

With some modifications, this quantity has potentially predictive value in STM imaging. In particular, STM barrier height images measure a quantity related to the average local ionization potential. The difference in the STM case is that not all states contribute to the sum. For a fixed, positive STM tip bias, only those sample donor states

---

[3]Koopmans, T. A., *Physica*, **1**, 104 (1933).

[4]Sjoberg, P.; Murray, J. S.; Brinck, T.; Politzer, P. *Can. J. Chem.*, **68** 1440 (1990).

[5]Brinck, T.; Murray, J. S.; Politzer, P.; Carter, R. E.; *J. Org. Chem.*, **56** 2934 (1991).

with energies equal to unoccupied states of the tip can contribute to tunneling (see Figure 1). Therefore, we can make a simple modification to Eq. (1.3) for the STM case

$$\overline{B}(r) = \sum_{a \ni \varepsilon_a > \varepsilon_{F_{tip}}}^{occupiedMOs} \frac{-\varepsilon_a \varphi_a^2(r)}{\sum_{b \ni \varepsilon_b > \varepsilon_{F_{tip}}}^{occupiedMOs} \varphi_b^2(r)} \tag{1.4}$$

where we have summed the contributing densities only over the states we presume to be available for tunneling. ($a \ni \varepsilon_a > \varepsilon_{F_{tip}}$ means select all states of the sample with orbital energies greater than the Fermi level of the tip.)

Orbital ionization potentials obtained by Koopmans' theorem tend to be too high. This is because we assume the same set of orbitals for the N-electron case as for the (N-1)-electron case. In reality, relaxation of the orbitals for the ionized species would result in an energy lowering, thus bringing the (N-1)-electron case closer to the N-electron case and lowering the ionization potential. This is somewhat offset by the fact that neglect of electron correlation in the Hartree-Fock wave function is generally more severe for the system with more electrons, but overall, the Koopmans' ionization potentials will be too high.[6]

Regardless of the inaccuracies in the absolute values of orbital energies, we may start at the model system's highest occupied molecular orbital (HOMO) and work our way down in energy collecting states that are potential electron donors in the STM system. For this window of states we can now plot a spatial map of high vs. low tunneling barrier heights. This map may be compared with the STM local barrier height images and constant height images.

---

[6]A. Szabo, N. S. Ostlund, **Modern Quantum Chemistry**, 1st Ed, revised, McGraw-Hill, New York, (1982). See p 123.

1.1.3.1. Average Local Electron Affinities

A related quantity of potential interest in STM imaging is the average local electron affinity. We define

$$\overline{EA}(r) = \sum_{a \ni \varepsilon_a < 0}^{unoccupiedMOs} \frac{-\varepsilon_a \varphi_a^2(r)}{\sum_{b \ni \varepsilon_b < 0}^{unoccupiedMOs} \varphi_b^2(r)} \tag{1.5}$$

where we have summed over the space of bound virtual orbitals in the Hartree-Fock wave function. This quantity describes the relative affinity of various regions near the molecule for accepting an electron. The absolute values of electron affinities for Hartree-Fock wave functions tend to be more in error than ionization potentials because the lack of relaxation of the (N+1)-electron state and the added correlation error due to the extra electron work in the same direction. Nevertheless, if we restrict our summation to an energy window of virtual states, we expect to get a map of the relative affinity of various regions for accepting an electron.

We have not yet obtained results for average local electron affinity calculations; however, we have presented the formalism for later use. Careful study must be done to determine the most appropriate way to calculate the virtual orbitals.[7]

1.1.4. Resonance Matrix Elements

In the section of this thesis on electron transfer between the STM tip and transition metal dichalcogenides, we used the methods developed by Voter and Goddard for

---

[7]The problem is that the Fock matrix elements for virtual orbitals have field terms for the full N electrons already in the occupied orbitals. Modifications can be made to the Fock matrix elements to improve the virtual orbitals by compensating for this.

evaluating Hamiltonian matrix elements between two non-orthogonal Hartree-Fock or GVB wave functions:[8]

$$H_{AB} = \langle \Psi_A | \hat{H} | \Psi_B \rangle$$

$$\Psi_A = A\left\{\left[\prod_{a=1}^{N_c} \varphi_a^A \varphi_a^A \alpha\beta\right]\left[\prod_{g=1}^{N_p}\left(\sum_{k=1}^{NO_g} c_{k,j}\varphi_{k,j}^A \varphi_{k,j}^A\right)\alpha\beta\right]\left[\prod_{s=1}^{N_o} \varphi_s^A \beta\right]\right\} \qquad (1.6)$$

$$\text{and similar for } \Psi_B \text{ but using a new set of orbitals } \{\varphi^B\}$$

where $\hat{H}$ is the electronic Hamiltonian for the N electrons in the antisymmetrized product wave functions Ψ_A and Ψ_B. N_C is the number of doubly occupied orbitals, N_p is the number of GVB expansions ("pairs" usually), NO_g is the number of natural orbitals for GVB "pair" g ($NO_g = 2$ usually), and N_O is the number of singly occupied orbitals. The $\{\varphi^A\}$ are orthonormal, as are the $\{\varphi^B\}$, but the overlap of orbitals between the sets are in general arbitrary. This leads to an $O(N!)$ number of terms to evaluate to complete the matrix element. The trick employed to solve this problem was to find a set of unitary transformations that biorthogonalize the two sets of orbitals to obtain two new sets $\{\varphi'^A\}$ and $\{\varphi'^B\}$ such that the overlap $\langle \varphi_a'^A | \varphi_b'^B \rangle = \lambda_{a,b}\delta_{a,b}$ where

$$\delta_{a,b} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise.} \end{cases}$$

This was possible because of the invariance of the energy to a unitary transformation of the occupied orbitals. For an excellent description of the details of this approach, see reference (8).

For the purposes of this work, we will use the matrix element calculations to obtain the electron transfer rates between two N-electron states describing a charge transfer system in the STM. Discussion of the specific models used appears in the section

---

[8]A. Voter, Ph. D. thesis, California Institute of Technology (1983).

on transition metal dichalcogenides. The chapter entitled "New Computational Approaches to Tunneling" presents more discussion of the advantages and disadvantages of this method for the prediction of electron transfer rates.

2. STM of Transition Metal Dichalcogenides

Reproduced here is an article summarizing our findings using a model system for tunneling from transition metal dichalcogenides to a tungsten STM tip. In images of these materials the current intensities associated with the positions of the underlying chalcogenide, transition metal, and crystal hollow form intersecting hexagonal lattices and are therefore geometrically indistinguishable. Therefore, any assignment of these peaks *must* rely on an underlying theoretical interpretation of the imaging mechanism.

The conclusion of primary interest here is that while the dark (low current) regions are indeed assignable to hollow sites, the assignment of chalcogenide or transition metal may depend upon the tip imaging height used. In particular, short tip-sample distances appear to image the surface (chalcogenide) atoms more strongly while long tip-sample distances appear to be dominated by the *second layer* transition metal atoms.

A secondary result of this work was that in all cases explored, the magnitude of the tunneling matrix elements mirrored the relative value of the density of the sample HOMO at the chosen tip height. Therefore, in subsequent work we focus on the computation of local densities at the Fermi energy and do not compute the more costly resonance matrix elements. We note, however, that the resonance matrix element approach can be very powerful if many-electron effects are important. Recently, we have re-implemented the resonance program with no internal limitations on the size of the wave function. The program also has the ability to be run on parallel computers.[9] We hope to soon test the new program for large wave function cases.

---

[9]Initial parallel tests will be conducted on a Silicon Graphics 8 processor computer. The next target is a 64 processor Kendall Square Research computer. The ability to run multiple determinant pairs (for GVB wave functions) simultaneously on multiple processors will lead to near linear speed ups because there is essentially no data dependency.

Theoretical interpretation of scanning tunneling microscopy images: Application to the molybdenum disulfide family of transition metal dichalcogenides

Terry R. Coley, William A. Goddard III, and John D. Baldeschwieler
Contribution No. 8184 from the Arthur Amos Noyes Laboratory of Chemical Physics, California Institute of Technology, Pasadena, California 91125

(Received 24 July 1990; accepted 7 November 1990)

We have performed *ab initio* quantum mechanical calculations to describe scanning tunneling microscopy (STM) images of MoS_2 and $MoTe_2$. These results indicate that the interpretation of the STM images of these and related materials depends sensitively on experimental conditions. For example, determining whether the maximum tunneling current correlates to the top atom (S or Te) or to the second-layer atom (Mo) requires information on the tip-sample separation. Based on these results we discuss some STM experimental procedures which would allow assignment of the chemical identity of STM spots with greater certainty.

I. INTRODUCTION

Scanning tunneling microscopy (STM) has clearly demonstrated its usefulness as a real-space atomic resolution surface probe. A striking example is the overwhelmingly conclusive evidence STM has provided for the correctness of the Takayanagi model[1] for the 7×7 reconstruction of the Si(111) surface.[2] In contrast to these successes, however, there remain many examples of STM images leading to ambiguous answers about the surface system in question. For example, many groups have imaged the transition.metal dichalcogenides,[3–6] but have disagreed in the interpretation of these images. The difficulty arises from the fact that the sample is heteroatomic, but the different elements reside in symmetry equivalent positions when projected onto a viewing plane parallel to the surface (see Fig. 1, top). Interpreting these images, therefore, requires a quantitative understanding of the electronic structure of the tip-sample system and the mechanism for electron transfer. As we will show, correct chemical assignment of the STM peaks in images of the MoS_2 family of compounds is one case where a detailed theoretical study is necessary to resolve ambiguities which arise due to the inherent symmetry of the system.

Because of their interesting electronic and mechanical properties,[7,8] the transition metal dichalcogenides have been the subject of numerous theoretical and experimental studies. Many of these properties arise from the layered structure of these materials. In particular, 2H – MoS_2, 2H – $MoSe_2$, and 2H–$MoTe_2$ consist of X–Mo–X (X = S,Se,Te) trilayers each with a central layer composed of hexagonally arranged Mo atoms. Each Mo has trigonal prismatic coordination to six X. See Fig. 1. The lack of covalent bonds between X–Mo–X sandwiches is a primary factor leading to the lubricative properties of MoS_2. This also allows easy cleavage to expose large areas of atomically flat MoX_2. These surfaces are sufficiently inert to be imaged by STM in air.[6]

Detailed information about the electronic structure of these surfaces is a prerequisite to predicting any useful materials modifications for attaching these compounds to substrates while retaining efficient lubricative properties. For this reason STM and scanning tunneling spectroscopy (STS) studies of the MoS_2 family of materials have been undertaken by many groups. However, attempts to interpret the results have been complicated by the high degree of symmetry present in the images: high quality fast-scan images (constant height) show three equivalent hexagonal lattices of high, medium, and low current.[5] The assignment of Mo, X, and crystal hollow to each of these current levels is there-

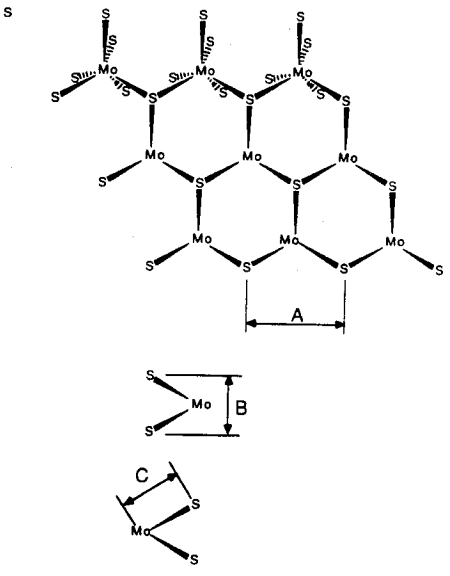


FIG. 1. Structure of the MoS_2 family of layered transition metal dichalcogenides (S = S or Te). Top: looking down on surface; first row of Mo indicates the trigonal prismatic coordination to the lower layer of chalcogenide. Bottom: cross section showing stacking of sandwich layers. Dimensions[21] for MoS_2: $A = 3.16$ Å, $B = 2.98$ Å, $C = 2.41$ A. Dimensions[22] for $MoTe_2$: $A = 3.52$ Å, $B = 3.63$ Å, $C = 2.72$ Å.

fore ambiguous without some assumptions about the electronic structure of the surface. Various authors have presented differing interpretations of these images with the concomitant differing arguments to justify them.[3–6] In summary, proximity to the tip argues for dominance of the chalcogenides, while the Mo d_{z^2} character of the states near the Fermi level argues for the metal's dominance.

Tang, Kasowski, and Parkinson[6] have in particular made progress experimentally with regard to correct assignment of the STM peaks. They have imaged WTe_2 which contains staggered (and therefore symmetry inequivalent) W atoms in the second plane. These images clearly showed dominance of the metal in the images. However, it was not clear if these results could be used unambiguously to predict that the peaks in their images of $MoTe_2$ were also due to the metal: uncertainty arises because W is octahedrally coordinated to Te while Mo is trigonal-prismatically coordinated. Furthermore, they performed an approximate calculation to determine the density of states above the surface. Their results indicated the spatial distribution of the density of states near the Fermi energy to be greatest over the Te atoms, which in the Tersoff and Hamann formalism[9] would indicate a greater tunneling current over the Te. Clearly, ambiguity still exists in the correct assignment of images in the MoS_2 family of materials. A major goal of this theoretical investigation is to provide firm ground for making interpretations of these images.

II. THEORETICAL MODEL

A. Tip-sample model

Models were developed specifically to describe the electronic structure of MoS_2 and $MoTe_2$ in the presence of a tip. Hartree–Fock wave functions for the tip-sample system were computed for various positions of the tip above the sample. For each geometry two predominant electronic states were studied: the first state described a hole (acceptor orbital) on the tip; the final state had the hole in the sample (one electron was transferred to the tip). For each tip-sample geometry, an all-valence-electron resonance calculation between the two states was performed.[10] The resulting transfer matrix element describes the amount of coupling between the two many-electron quantum states. Differences in the transfer probability as a function of tip position reflect the overall changes in tunneling current expected from a true tip-sample system with bulk orbitals composed of a linear combination of the cluster states as a basis.

B. Model for the MoX_2 bulk

The particular clusters used for MoS_2 and $MoTe_2$ consist of seven Mo atoms arranged in a centered hexagon with the six chalcogenide atoms arranged in a trigonal prism. Figure 2 shows the geometry of this cluster. For the central Mo we used Hay and Wadt's[11] valence double-zeta basis set and effective core potential (ECP). This basis was contracted to a minimum basis set for the six external Mo atoms. The chalcogenides were also treated at the valence double-zeta level. The Hay and Wadt basis and ECP were used for Te, and the Rappe, *et al.*[12] basis and ECP were used for S.

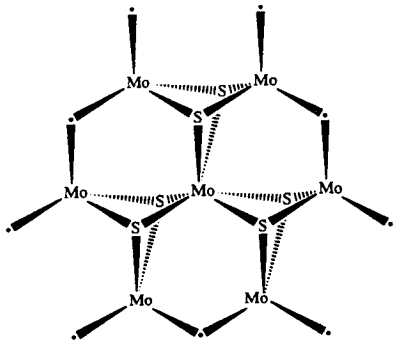


FIG. 2. Structure of the cluster to model MoS_2 and $MoTe_2$ (S = S or Te). The positions of all atoms are taken from the bulk structure (see Fig. 1). Dots represent point charges which replace the chalcogenide nuclei at the other end of the dangling bonds. Only nine point charges are shown; nine more exist in the second chalcogenide plane.

Much attention was paid to designing this cluster to mimic the bulk system. The cluster gives the proper chemical bonding environment to the central MoX_6 trigonal prism: all bonds are satisfied as in the bulk. The treatment of the exterior Mo atoms was constructed so that the Mo *d* electrons were distributed in as many orbitals as would exist if the exterior Mo atoms were fully contained in the bulk. Each Mo has six valence electrons, two of which end up in a nonbonding d_{z^2} orbital. The other four can be thought of as contributing equally to six Mo–X bonds, 2/3 electron to each bond. Therefore, since each Mo of the exterior ring has four broken bonds and a d_{z^2} orbital, 28 electrons should occupy the 30 orbitals representing the dangling bonds and nonbonding orbitals at the cluster edge. However, we require an integral number of electrons per orbital (we cannot use an effective Hamiltonian) in the cluster for calculation of many-electron matrix elements. Thus, we added two electrons resulting in an overall charge of -2. The electrons in these orbitals were coupled high spin to prevent back bonding to the center of the cluster. The charge of the cluster was also made neutral by adding 18 counter charges of $+0.11e$ placed at the positions of the chalcogenide nuclei that the exterior Mo atoms would be bonded to. This further served to direct the exterior *d* electrons toward the missing centers they would normally be bonded to. The result of this work is a chemically well described core trigonal prism of material.

C. Quality tests for the bulk model

We found Mo_7X_6 to be the minimum cluster necessary to give a good description of the surface. Attempts to use a naked trigonal prism (MoS_6) without the surrounding ring of Mo resulted in an ordering of molecular orbital (MO) states which did not reflect the bulk band structure.[11] This was the result of inadequate lowering of the chalcogenide *p*-orbital energies when the full bonding environment to Mo was not established. Results with MoS_6 clusters gave ionization potentials for the surface of 7–8 eV while a neutral

Mo_7S_6 cluster gives 5.2 eV, much closer to the experimental bulk work function of 4.90 ± 0.15 eV.[7]

Because the basis set consists of Gaussian functions, we were careful to ensure that the resulting wave functions behaved with proper exponential decay in the region between tip and sample. Figure 3 shows a log plot of the wave function amplitude for the highest energy occupied MO (HOMO) of the tip-sample system which is primarily Mo d_{z^2} in character. The plot shows the expected exponential decay in the vacuum region.

As an additional test of the quality of our wave function, we augmented the basis set on the S directly below the tip atom in the Mo_7S_6 cluster to contain two additional diffuse functions.[14] The most diffuse of these functions had a radial exponent which provided a slower falloff into the vacuum than the Mo functions. Even with diffuse functions on the sulfur, the quantitative results were modified by 20% at most and the qualitative conclusions we will draw remain the same.

In summary, we have shown that Mo_7X_6 is a carefully crafted, successful model for the MoX_2 bulk. It has the advantage of being a wholely self-consistent all-valence electron description of a chemically relevant section of the crystal.

D. The tip

The STM system also contains a tip modeled by a Ba atom. Ba was chosen because its ionization potential of 5.2 eV closely matches the bulk work function for tungsten. W is used as the tip material for many STM experiments, including those on MoX_2. In our model the s orbital of the Ba atom will serve as the accepting orbital present at the end of the STM tip.

E. Tunneling current

As mentioned earlier, two states are calculated: Ψ_A is the self-consistent N-electron wave function having a hole (acceptor) on the tip while Ψ_B is the self-consistent wave function having a hole in the sample. The matrix element describing the tunneling probability for a two state system is

$$T_{BA} = (H_{BA} - S_{AB}H_{AA})/(1 - |S_{AB}|^2),$$

where

$$H_{BA} = \langle \Psi_B | H | \Psi_A \rangle$$

$$H_{AA} = \langle \Psi_A | H | \Psi_A \rangle$$

$$S_{AB} = \langle \Psi_A | \Psi_B \rangle,$$

and H is the full N-electron Hamiltonian. Since Ψ_A and Ψ_B are the fully relaxed many-electron wave functions, these calculations include the response of the other N-1 electrons to the electron transfer process.

The use of T_{BA} to describe electron transfer rates is well established in the chemical literature.[15,16] Its relationship to solid-state electron transfer theory is clearly described in West *et al.*[17] and is derived from Bardeen's time-dependent, first-order perturbation theory.[18] It is important to note that for each matrix element above, the full valence electron Hamiltonian of the system is evaluated with respect to the determinantal wave functions Ψ_A and Ψ_B. Ultimately, the matrix element calculations reduce to evaluating integrals where the integrand contains Gaussian functions (from the basis set expansion of the wave function) centered at various atomic positions and operated on by the one and two electron terms in the Hamiltonian. Such calculations are ubiquitous in modern molecular quantum chemistry,[19] however, evaluation of the cross matrix element H_{AB} between two nonorthogonal, N-electron, determinantal wave functions requires a prior biorthogonalization procedure in order to avoid $N!$ expansion of the time for evaluation of these matrix elements.[10]

The theory above is derived in the small bias voltage limit. In fact, the above formalism describes electron transfer between just two states of the tip and sample. In practice, as the bias voltage is increased many more states are sampled for electron transfer. We consider one band of states, the valence d_{z^2} band. To calculate the total experimental tunneling current, it would be necessary to sum the contributions from all states in the d_{z^2} band derived from our single lattice site basis. We do not do that here, but rather focus on the intuition gained from studying the underlying coupling constants between states of the tip and sample in a localized region above the surface. An additional effect of the bias voltage is that a strong field is generated between the tip and sample. If the surface state is sufficiently polarizable, this could modify the shape of the surface wave function enough to change the qualitative peaks in the tunneling current as a function of position. We tested this case by generating a new set of tip-sample system wave functions converged under a field corresponding to tip-positive by 1 V. Although, the relative energies of states in the bulk change slightly, there was no effect on the relative current as a function of tip position from that predicted by the matrix elements at close to zero field which we present here.

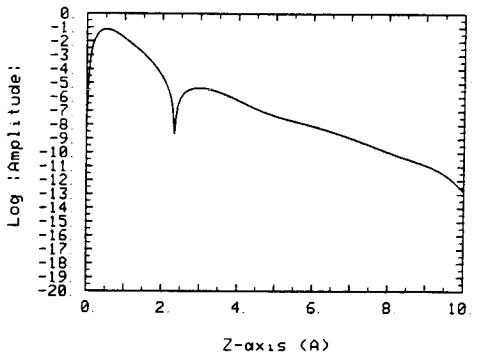


FIG. 3. Log plot of the absolute amplitude of the HOMO (donor orbital) in the $MoS_2 \cdots$ Ba sample-tip system. The amplitude is calculated along a line perpendicular to the surface and passing through the central Mo and the Ba. The Mo plane is at $z = 0$ Å, the S plane is at $z = 1.49$ Å, and the Ba tip is at $z = 10.0$ Å.

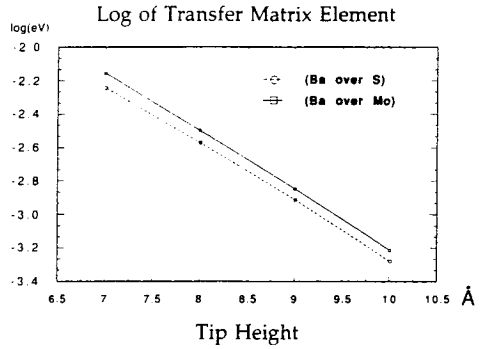


FIG. 4. Log plot of T_{B1} for the $MoS_2 \cdots$ Ba sample-tip system as a function of tip atom distance over the Mo plane (at 0 Å). For the solid line, the tip was over the central cluster Mo. For the dashed lined, the tip was over a cluster S. The units of T_{B1} were eV.

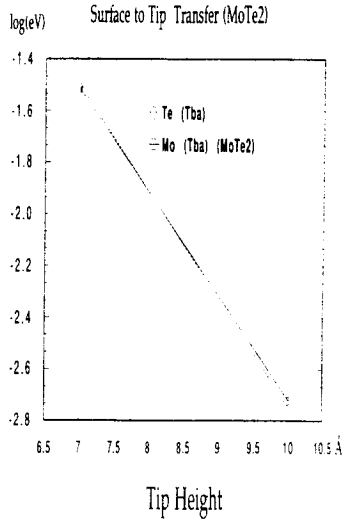


FIG. 5. Log plot of T_{B1} for the $MoTe_2 \cdots$ Ba sample-tip system as a function of tip atom distance over the Mo plane (at 0 Å). For the solid line, the tip was over the central cluster Mo. For the dashed lined, the tip was over a cluster Te. The units of T_{B1} were eV.

III. RESULTS SUMMARY

A. MoS_2

Figure 4 shows a log plot of the transfer matrix element T_{BA} at four tip-sample separations and at two lateral positions above the surface. The solid line shows the behavior of T_{BA} while the tip atom is over the central Mo, the dashed shows T_{BA} with the tip directly above a sulfur. We see that the electronic coupling T_{BA} is larger with tip over the Mo for all tip-sample separations explored. We also see an exponential decay of T_{BA} as expected from the exponential decay of the wave function.

B. $MoTe_2$

Figure 5 shows a plot for $MoTe_2$ similar to that for MoS_2. Unlike the MoS_2 plot no clear trend of dominance appears for electronic coupling over the Mo versus the Te. The calculations suggest that the coupling magnitude *may* cross at some tip-sample separation.

IV. DISCUSSION

A. Comparison of MoS_2 and $MoTe_2$

Comparison of MoS_2 and $MoTe_2$ shows an overall greater coupling between $MoTe_2$ and the tip at corresponding tip heights. This is to be expected from the more diffuse Te p orbitals and lower ionization potential. Thus, for the same current set point we expect the tip to glide higher over $MoTe_2$ than MoS_2.

B. Predictions for $MoSe_2$

Based on the trends for MoS_2 and $MoTe_2$ we expect $MoSe_2$ to couple with some intermediate strength to a tip. As in MoS_2, coupling will dominate while the tip is over Mo. This is based on the clear Mo dominance in MoS_2 and the nearly flat contrast seen for $MoTe_2$.

C. Experimental consequences

Our results suggest that based on electronic effects alone, there is good reason to believe that it is the second-layer Mo sites which give rise to the largest current in tip-positive STM images of MoS_2. Our results predict this trend will persist for even closer tip-sample distances, than those calculated. However, at very close distances the form of the surface wave function suggests a reversal of the image-dominant atom from Mo to S will occur. This is due to the presence of roughly 27% sulfur p character in the HOMO. At present, we do not know if this will occur before breakdown of gap resistance. As described next, we have proposed an experiment to resolve these tip-height issues.

The theoretical results for $MoTe_2$ show the electronic surface in the region from 5.5 to 8.5 Å above the Te is quite planar; thus, little corrugation is expected from variations in tunneling probability. One possible explanation for the corrugation observed by Tang, Kasowski, and Parkinson[6] might involve a mechanism like that proposed by Zheng and Tsong[20] for atomic resolution images of metal surfaces. This mechanism relies on the presence of an impurity on the tip with appropriate energy states to allow a resonance tunneling effect which is then modulated by the changing position of the impurity due to its response to the forces it experiences while trapped between the tip and sample. A simpler explanation for the $MoTe_2$ system brought to light by our calculations is that the tip is *closer* than 5.5 Å from the surface so that the electronic differences between Mo and Te and their differing distances from the tip are exaggerated. Since the primary character of the states near the Fermi level is Mo d_{z^2}, the tip could approach much closer to the Te surface without establishing a conductive contact (loss of gap resistance). In this case the Te layer simply serves to modulate the underlying donor states in the Mo d_{z^2} band. We plan to explore closer distances with further detailed calculations.

Based on our results for $MoTe_2$ we have also devised a new STM experiment in collaboration with Youngquist, Driscoll, and Baldeschwieler which should help distinguish between possible imaging mechanisms and provide information needed to make a definite chemical assignment. The experiment is designed to collect images at multiple tip heights free of lateral drift. We hope to show experimental evidence for the possible crossover of Mo versus Te dominance in images taken at different heights. Establishing the existence or absence of this crossover is very important to correct interpretations of these images: if the chemical assignment is tip-height dependent, careful probing of in-phase images at multiple heights will become a prerequisite to the correct chemical assignment of peaks in STM images of these and other materials.

V. ACKNOWLEDGMENTS

This research was supported in part by a grant (No. AFOSR-88-0051) from the Air Force Office of Scientific Research and the Office of Naval Research (N00014-86-K-0214). Preliminary studies were carried out earlier with funding from Hughes Aircraft Company (Mike Gardos). Additional funding came from the National Institutes of Health (R01 GM 37226-03). Calculations were performed on Alliant FX/8-8 and FPS Computing Model 511 minisupercomputers.

[1] K. Takayanagi, Y. Tanishiro, M. Takahashi, and S. Takahashi, J. Vac. Sci. Technol. A **3**, 1502 (1985).
[2] G. Binnig, H. Rohrer, Ch. Gerber, and E. Weibel, Phys. Rev. Lett. **50**, 120 (1983).
[3] H. Bando, H. Tokumoto, W. Mizutani, K. Watanabe, M. Okano, M. Ono, H. Murakami, S. Okayama, Y. Ono, S. Wakiyama, F. Sakai, K. Endo, and K. Kajimura, Jpn. J. Appl. Phys. **26**, L41 (1987).
[4] G. Stupian and M. Leung, Appl. Phys. Lett. **51**, 1560 (1987).
[5] M. Weimer, J. Kramar, C Bai, and J. D Baldeschwieler, Phys. Rev. B **37**, 4292 (1988).
[6] S. L. Tang, R. V. Kasowski, and B. A. Parkinson, Phys. Rev. B **39**, 9987 (1989).
[7] J. C. McMenamin and W. E. Spicer, Phys. Rev. B **16**, 5474 (1977).
[8] P. D. Fleischauer, Thin Solid Films **154**, 309 (1987).
[9] J. Tersoff, and D. R. Hamann, Phys. Rev. Lett. **50**, 1998 (1983).
[10] A. F. Voter, Ph.D. thesis, California Institute of Technology, Pasadena, 1983. (Further improvements by J-M. Langlois).
[11] P. J. Hay and W. R. Wadt, J. Chem. Phys. **82**, 270 (1985).
[12] A. K. Rappe, T. A. Smedley, and W. A. Goddard III, J. Phys. Chem. **85**, 1662 (1981).
[13] R. Coehoorn, C. Haas, and R. A. de Groot, Phys. Rev. B **35**, 6203 (1987).
[14] Two additional *s* and two additional *p* exponents were scaled out from those of Ref. 12. *s* function: 0.0499, 0.0164; *p* function: 0.0435, 0.0125.
[15] N. R. Kestner, J. Logan, and J. Jortner, J. Phys. Chem. **78**, 2148 (1974).
[16] J. Logan, and M. D. Newton, J. Chem. Phys. **78**, 4086 (1983).
[17] P. West, J. Kramar, D. V. Baxter, R. J. Cave, and J. D. Baldeschwieler, IBM J. Res. Develop. **30**, 484 (1986).
[18] J. Bardeen, Phys. Rev. Lett. **6**, 57 (1961).
[19] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, 1st ed. (Macmillan, New York, 1982).
[20] N. J. Zheng, and I. S. T. Tsong, Phys. Rev. B **41**, 2671 (1980).
[21] R. W. G. Wyckoff, *Crystal Structures*, 2nd ed. (Wiley, New York, 1963), Vol. 1, p. 280.
[22] D. Puotinen and R. E. Newnham, Acta. Crystallogr. **14**, 691 (1961).

3. STM of Small Adsorbates on Ni (110)

3.1. Introduction

To the surprise of many in the STM community, Eigler and Schweitzer[10] were able in 1989 to image single Xenon atoms on a clean Ni (110) surface. The paradox of this system is that the electronic structure of the sample-adsorbate system might be expected to render the Xe atoms invisible. In particular, the localized states of the adsorbate appear well below (HOMO) and well above (LUMO) the Fermi energy of the metal. It would seem that these states are incapable of significantly contributing to the STM tunneling current since the experiments were carried out at a relatively low bias voltage (e.g., 20 mV), and thus the localized atomic images were something of a mystery.

Much discussion has surrounded the nature of the mechanism for STM imaging of adsorbates. Some early proposals relied on the presence of adsorbate states near the Fermi energy of the surface. Often, such states do not exist, especially for small, low atomic number molecules. For most isolated, small molecule adsorbate systems the adsorbate HOMO and LUMO lie well below and above the metal substrate Fermi energy. Proposals which postulated adsorbate energy level shifts due to strong interactions with the tip[11] had to be ruled out in the Xe/Ni case. Here, it was clearly demonstrated that the tip could be only weakly interacting with the adsorbate; otherwise the Xe atoms would always be moved by the tip.

Another possibility is that the adsorbate changes the total density of states at the Fermi energy. This effect is most pronounced in systems of high coverage where a

---

[10]D. M. Eigler, E. K. Schweizer, *Nature* **344** 524 (1990).

[11]R. Garcia, N. Garcia, *Chem. Phys. Lett.* **173** 44 (1990).

molecule is chemisorbed on the surface. In this situation, surface states are tied up in bonding with the adsorbate, thus pulling the bonding states below the Fermi energy and pushing antibonding combinations above the Fermi energy. It is not the active mechanism for physisorbed species where the interaction between substrate and adsorbate is weak. Depending on the STM tip bias chosen, chemisorbed species can result in a darkening, or lightening of the STM spots near the adsorbate. These effects also occur near strongly electronegative elements on the surface.[12]

In this section we develop an intuitive picture of the mechanism responsible for STM imaging of small molecules physisorbed on metal surfaces. Our model remains consistent with the expected electronic structure of these systems and it is valid for many similar STM-surface-adsorbate systems. The imaging mechanism is quite simply due to the Pauli principle: where the metal wave function tails pass through the adsorbate states, the two must become orthogonal. The least energetically costly way to do this is to introduce oscillations in the tail of the metal wave functions. These oscillations are responsible for the modulation of the tunneling probability that give rise to the STM images.

To demonstrate this effect we have performed *ab initio* quantum mechanical calculations that reproduce the STM images of Xe on Ni (110) as obtained by Eigler and Schweitzer. We also present results for the expected image of a hypothetical cyclopropene molecule on Ni (110). In what follows we refer to several basis sets by name. These are detailed in the appendix to this chapter.

---

[12]E. Kopatzki, R. J. Behm, *Surface Science*, **245** 255-262 (1991).

3.2. Xe on Ni (110)

3.2.1. Imaging Mechanism

SYNOPSIS:

We show visually the extended oscillations in the Ni HOMO that only appear when a Xe atom is present. The result is a chemically intuitive picture demonstrating the mechanism for STM imaging of Xe on Ni (110).

As states at the Fermi energy of the substrate pass through the adsorbate, the behavior is no longer that of exponential decay, but of rapid amplitude oscillation. This oscillation delays reduction of the surface wave function amplitude near the adsorbate. As a result, the STM tip, which senses the tail of the wave function, traces a new, extended path past the adsorbate. This does not require the existence of localized adsorbate states near the Fermi energy. The extra oscillations are built in automatically to the *surface wave function* by the Pauli exclusion principle.

The best way to visualize what is producing the STM images is to present a three-dimensional view of the Ni HOMO with and without the Xe atom present, as in Figures 2 and 3. Figure 2 shows the cluster approximation to a plane wave decaying into the vacuum. Figure 3 shows the combination of Xe s and p_z functions being used by the Ni HOMO to become orthogonal to the other states of the Xe atom. The energy cost of mixing in small amounts of Xe states is low because the affected region is only a small portion of the total density of the Ni wave function. However, the result is a physical extension of the Ni HOMO further into the vacuum. In the following sections we will examine this result in more detail.

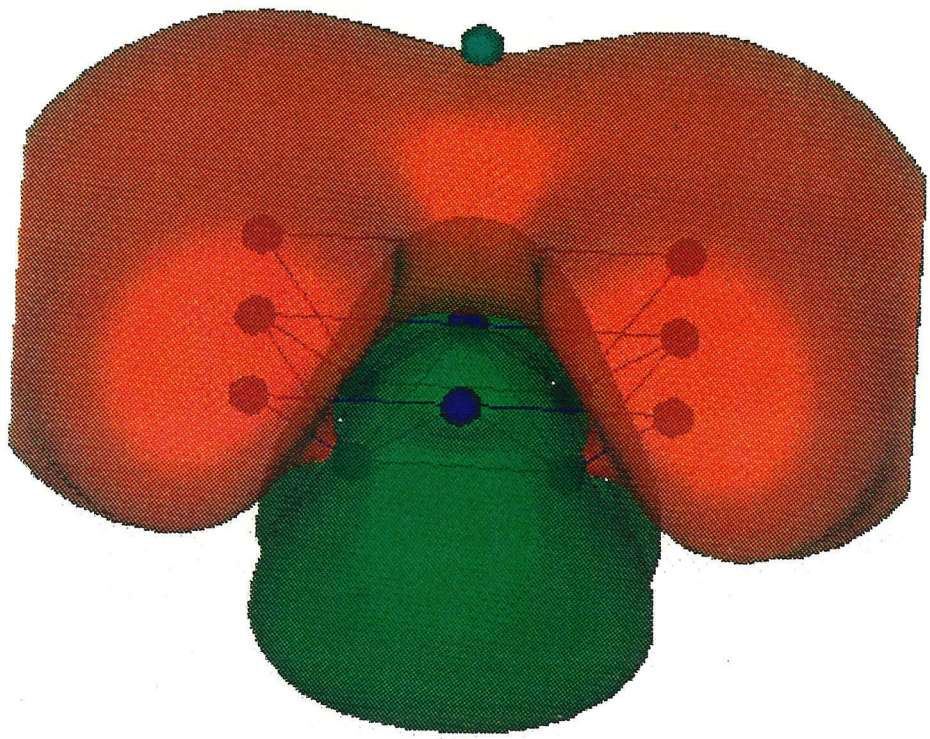


Figure 2: Xe/Ni_{13} HOMO with no actual Xe atom present. The amplitude isosurface is 0.005 $bohr^{-3/2}$ with phase indicated by color.
The location of the Xe nucleus shown for reference with Figure 3.
(Basis: Ni DZP1 / dummy functions at Xe center: Xe DZ + DIFs)

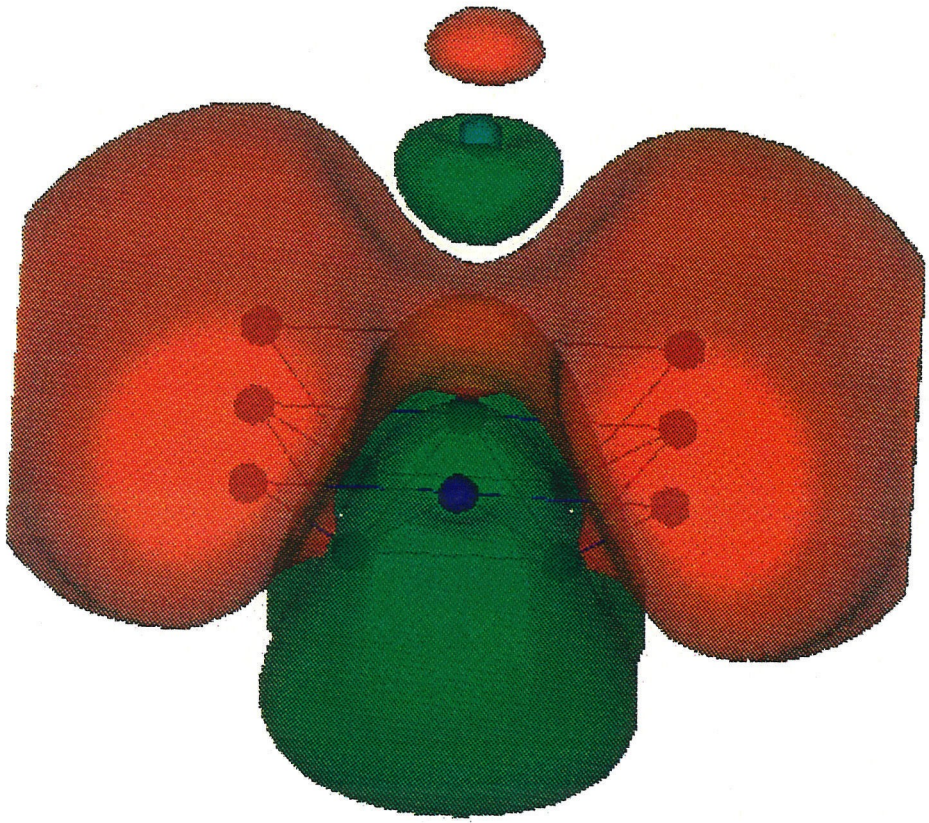


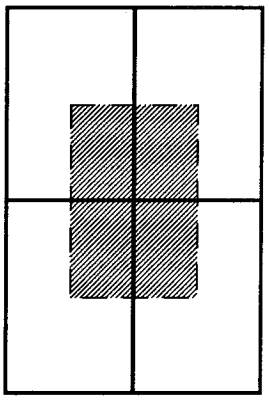
Figure 3:
Xe/Ni_{13} showing the perturbative affect of the Xe on the Ni HOMO.
All plot parameters are as in Figure 2.
(Basis: Ni DZP1 / Xe DZ + DIFs)

3.2.2. Apparent Height of Xe on Ni

SUMMARY:

We can make a rough estimate for the retraction of a point-tip while scanning at constant current over Xe deposited on Ni. Our estimate of 2.2 Å is compared to the experimental result of 1.5 Å.

Our model allows us to make an estimate of the constant current mode tip retraction over Xe. The following shows a top view of the Ni_{13} cluster. Ni atoms are located at each vertex.



The shaded patch represents the shadow of a plane placed above the Ni surface. We locate the plane at a plausible tunneling tip distance of 6.8 Å above the surface (twice the distance of the Xe atom). Next, we compute the Ni HOMO amplitudes on this patch plane for the cluster with Xe present using the Ni DZP1 / Xe DZ+DIFs basis set. We find the range of amplitudes to be from 0.00046 $bohr^{-3/2}$ to 0.00077 $bohr^{-3/2}$. Now, using the cluster with Xe removed to infinity (actually, +100Å for calculational convenience), we compute the amplitudes for the same Ni orbital in the same patch until at different heights we find a distance above the surface that generates comparable amplitudes. This height is

4.63 Å and has an amplitude range of 0.00041 $bohr^{-3/2}$ to 0.00077 $bohr^{-3/2}$. Thus we predict a tip retraction (apparent Xe atom height) of roughly 2.2 Å for constant current mode tunneling. Here, we have not accounted for finite tip size effects which will tend to blur the image and reduce the measured height of the Xe. Later, we explore finite tip effects.

Eigler *et al.*,[13] report a tip retraction of 1.53 +/- 0.02 Å over the Xe atoms. Thus, we find our calculations yield reasonable results. In the experiment there is no direct way to measure an absolute tip-sample distance. Using crude estimates, Eigler suggests a tip-sample distance of 13 bohr (6.9 Å). Note that the apparent height of the Xe will depend on the distance above the Ni surface used to evaluate the amplitudes. The farther the tip, the more pronounced the retraction over the Xe (larger apparent height). This somewhat non-intuitive result is due to the fact that the contribution of the Xe excited state falls off more slowly than the Ni states because the Ni states originate from 4*s* orbitals while the contributing Xe orbitals are more like Cs 6*s* orbitals. The enhanced height differential expected does not lead to improved resolution. On the contrary, lateral resolution is lost at larger tip-sample distances due to the spherical nature of the Xe *s*-tail which spreads out laterally at larger distances.

In conclusion, although we could find a tip-sample distance (possibly around 6.9 Å) that results in a match between our apparent Xe height and the experiment, we should not over burden the level of calculation we have done. We stress that these calculations are intended to give a qualitative picture of the approximate relative current levels over Xe and Ni.

---

[13]D. M. Eigler, P. S. Weiss, E. K. Schweizer, N. D. Lang, *Phys. Rev. Lett.*, **66** 1189 (1991).

3.2.3. Apparent Barrier Heights Over Ni

SYNOPSIS:

We compute STM effective barrier heights over Ni and Xe to be 0.53 and 0.92 eV, respectively. We predict that barrier height images of this system would yield high barriers at topographic peaks. These values also imply that measured Ni-Xe separations are necessarily less than their true value.

In Figures 4 and 5 we have plotted fits to the Ni HOMO tails with and without a Xe atom present. The STM effective barrier heights may be computed using (1.2) using the fit parameters in the Figures. For the bare Ni surface $\phi_{eff} = 0.53$ eV while over the Xe atom the value becomes $\phi_{eff} = 0.92$ eV. These values are in accord with effective barrier height measurements that have been performed on other STM systems[14] and should not be confused with atomic ionization potentials or bulk work functions: the values are derived in the context of a simple one-dimensional tunneling model.[1] The *ratio* of these values does, however, fall between the ratios of the ionization potential of Xe (12 eV) to Ni atom (7.6 eV) and Xe atom to Ni bulk work function (roughly 4.8 eV), as we would expect.

These values predict maxima in the STM barrier height images at the peaks of the topological or fast-scan images. Barrier height images correlated with topology or fast-scans have not been performed on this system. However, this somewhat non-intuitive result is consist with actual experimental results on other systems and is discussed in later sections.[30] The results also imply that the tip passes closer to the Xe than its average value over the Ni surface. Therefore, STM measurements of the Ni-Xe separation are smaller than the actual values.

---

[14]J. K. Gimzewski, R. Möller *Phys. Rev. B* **36** 1284 (1987).

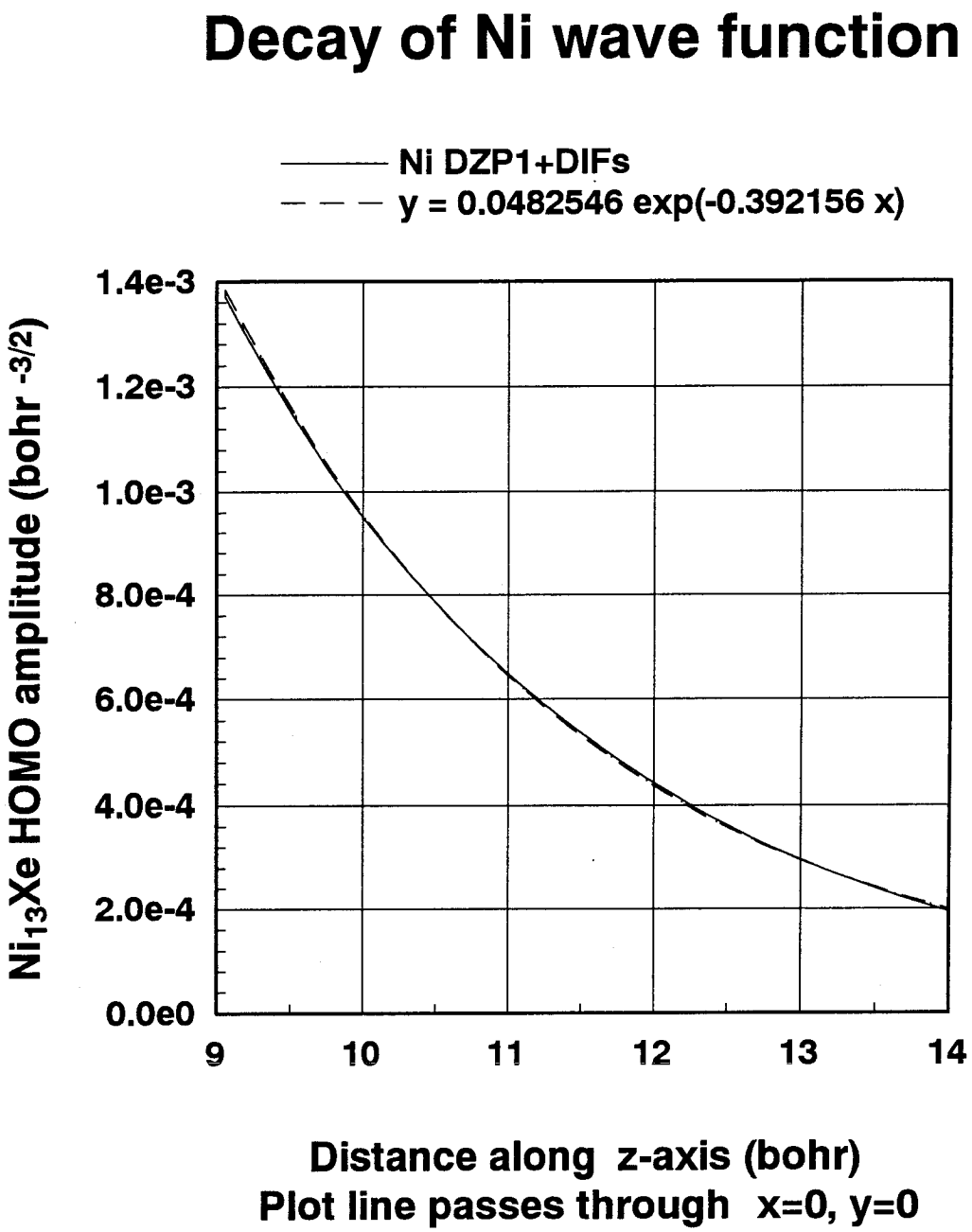


Figure 4:

Decay of the Ni HOMO over a range from 9 to 14 bohr above the Ni surface shown with an exponential fit.

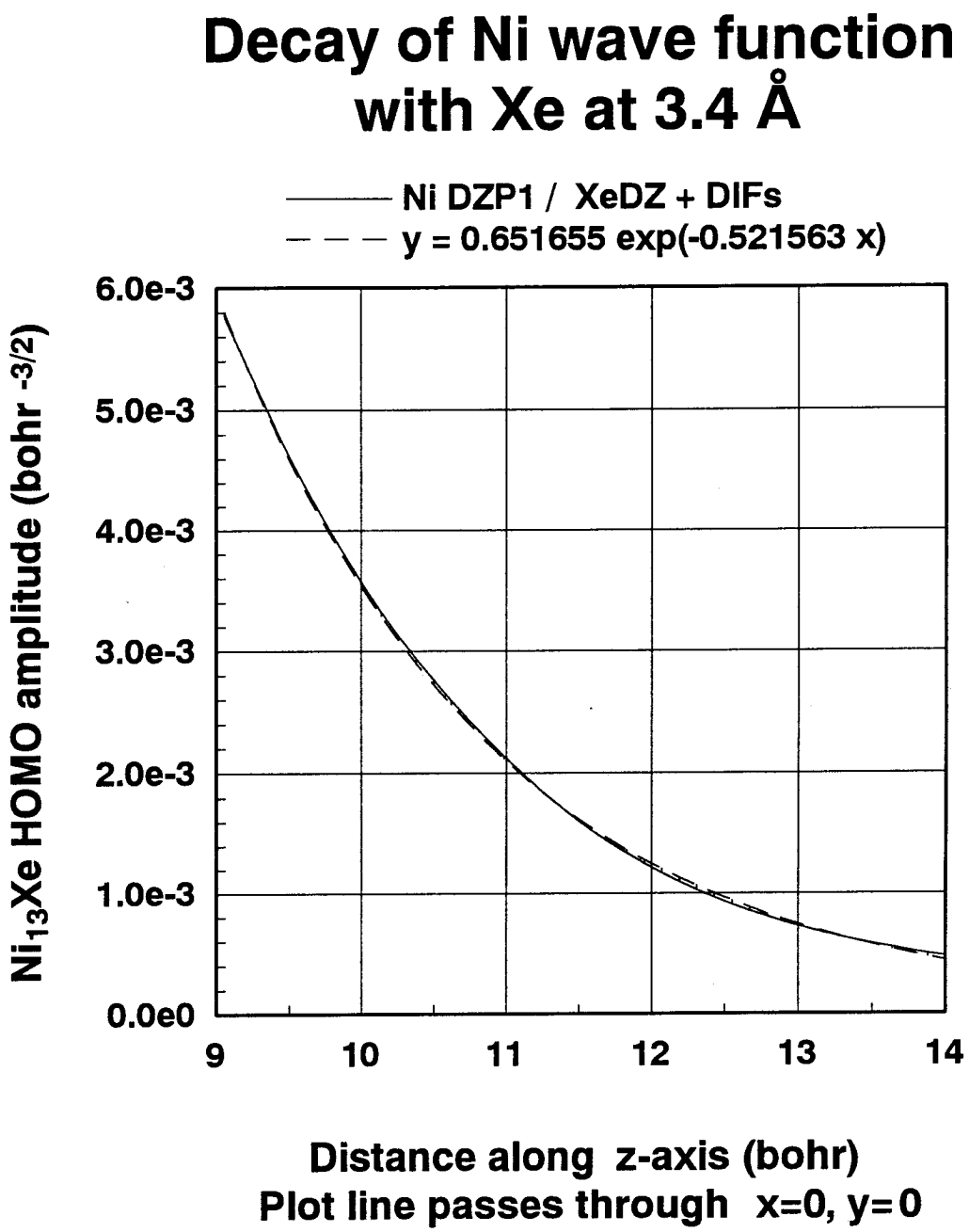


Figure 5:

Analogous to Figure 4 but with a single Xe atom placed at z = 3.4 Å (6.425 bohr).

3.2.4. Decay Through a Series of Non-Bonded Adsorbates

SYNOPSIS:

We show that the presence of a collection of non-bonded atoms located between the tip and sample has the effect of greatly reducing the decay effective constant of the substrate wave function relative to its value for vacuum decay. The reduction in the decay constant of the surface wave function is a function of the compression of the non-bonded states in the tip-sample gap.

We have seen how the presence of a Xe atom acts as a medium that modulates the normal decay of the Ni states into the vacuum. Here, we examine how the presence of additional Xe atoms affects the Ni HOMO decay at still further distances. This model has bearing on STM experiments where tunneling occurs through more than one non-bonded region of an adsorbate molecule. The basis set used in this section was Ni DZP1 / Xe DZ + DIFs.

The minimum in the Xe-Xe potential is near 4.3 Å[21] and so we positioned Xe atoms at 3.4 Å (Xe on-top Ni), 7.7 Å, 12.0 Å, and 16.3 Å and recomputed the cluster wave function. Figure 6 shows a blow up of the Ni HOMO as it decays through the four Xe atoms. As can be seen, oscillations emanating from the Ni surface appear as additional character from the *s* and *p*-functions of each subsequent Xe. In Figure 7 a fit to the upper peaks of these oscillations is shown. The effective decay constant (-0.28 $bohr^{-1}$) can be compared with that of Ni (110) with Xe at 3.4 Å (-0.52 $bohr^{-1}$) and that for the free Ni (110) cluster (-0.39 $bohr^{-1}$). The presence of the intervening Xe atoms substantially extends the Ni HOMO.

Although it is certainly not the case in the IBM Xe on Ni experiment (see below), there is evidence that the STM tip can place significant forces on the molecule being

imaged when the adsorbate is pinned down, either by a chemical bond or packing forces.[15] Typical forces discussed are from 10^{-10} to 10^{-8} N. We examined the result of compressing the chain of Xe atoms by making each one 3.4 Å from the previous one. To give an idea of the force needed to do this, we did a rapid calculation at the Hartree-Fock level of a Xe dimer at 3.4 and 4.3 Å. The energies are -30.43126 hartree and -30.44775 hartree, respectively, for a ΔE of 0.016486 hartree. For the distance change of 1.1 Å this corresponds to a force of 0.65 nN to do the compression of each pairwise interaction. This also predicts an energy difference of 0.049458 hartree between the cluster with Xe separated by 4.3 Å and the cluster with Xe separated by 3.4 Å (three pairwise compressions take place). In fact the energy difference between the clusters is 0.049995, in accord with our estimate. So the total force to achieve the compression of a chain of n non-bonded atoms like Xe-Xe can be estimated at n-1 times the pairwise force. For our cluster, the force needed would be 1.95 nN exerted by the tip toward the sample.

Figure 8 shows the decay of the Ni HOMO through the compressed chain, and Figure 9 shows the effective decay constant has been reduced to -0.23 $bohr^{-1}$. As a result, the amplitude of the Ni HOMO at the last peak due to Xe is about 10 times larger in magnitude for the compressed chain medium than it is for the uncompressed chain. Therefore, the Ni HOMO *density* at the location of a tip is larger by about a factor of 100.

From these results we can conclude that in cases where the adsorbate imaging mechanism is a perturbation of the substrate wave function via interacting non-bonded units, tip pressure on the adsorbate can indeed have a significant effect. Of course, for the tip to apply this pressure the adsorbate must be held stationary by a greater lateral force. In the case of the IBM Xe on Ni experiment such forces could not have been active. The

---

[15]U. Dürig, J. K. Gimzewski, D. W. Pohl, *Phys. Rev. Lett.* **57** 2403 (1986).

diffusion barrier for Xe on Pt (111) is roughly 30 meV.[16] For a motion of roughly 2 Å from the on-top site to a bridge site this corresponds to a force of 0.024 nN. Assuming a similar value for Xe/Ni (110), any such tip pressure capable of squeezing non-bonded interactions in the system would also push the Xe along the surface. It is a testimonial to the sensitivity of the IBM STM operating at 4°K that they were able to at will either move the Xe atoms, or image without disturbing the Xe atoms.

Such tip pressures on the adsorbate-sample system may not be ruled out in other systems. For example, when liquid crystals and other organic molecules that form packed, organized layers on the substrate are imaged, the adsorbates are held in place by the packing. Here, perturbation of the substrate wave function may be enhanced by the tip pressing the adsorbate into the substrate. However, as we will see later, unless small bias voltages are used, liquid crystal imaging is most likely via adsorbate states directly.

---

[16]K. Kern, R. David, P. Zeppenfeld, G. Comsa, *Surf. Sci.* **195** 353 (1988).

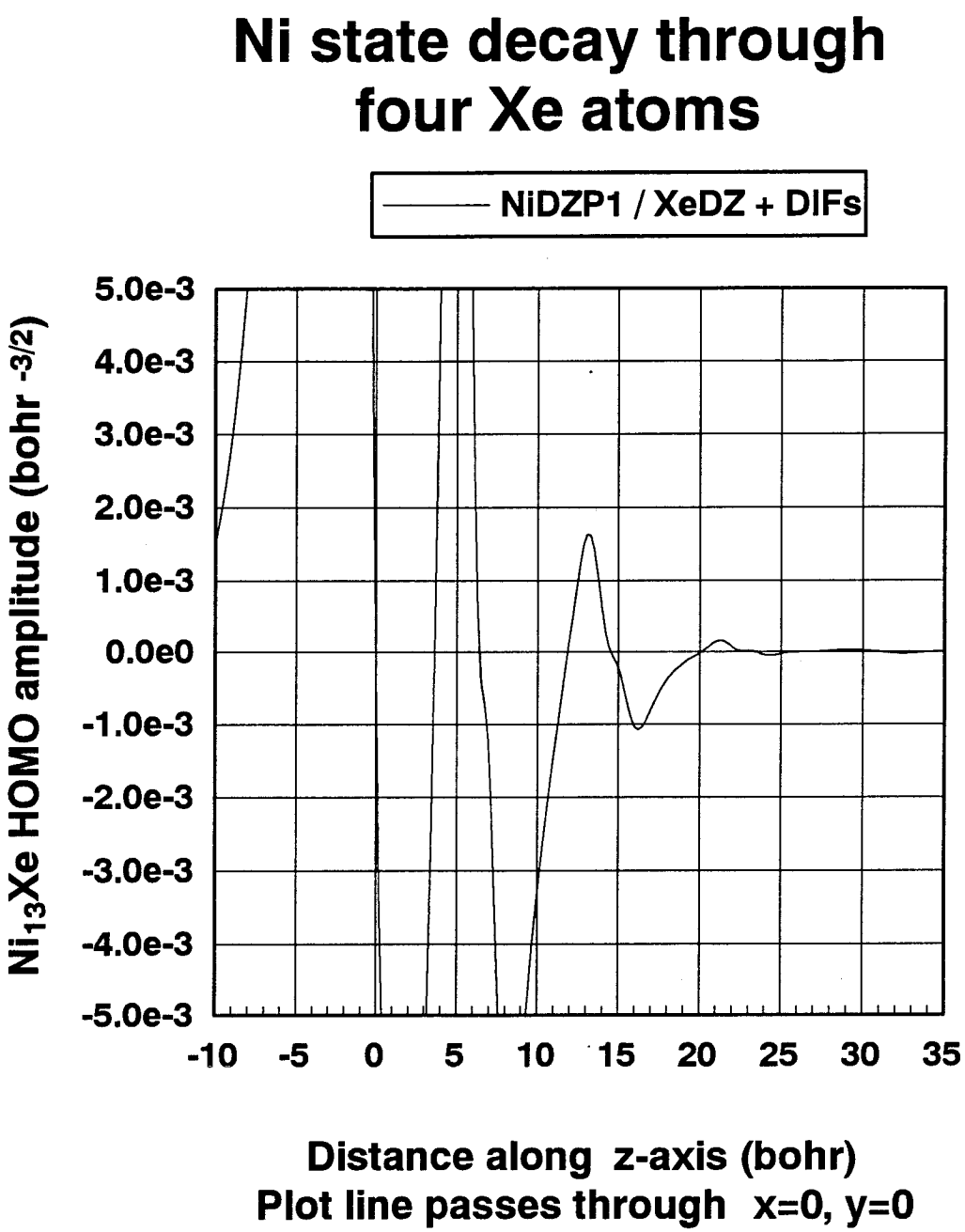


Figure 6: Decay of Ni HOMO along line x=y=0 through four Xe atoms located at 3.4, 7.7, 12.0, and 16.3 Å (6.4, 14.6, 22.7, and 30.8 bohr). The Ni subsurface layer is at z=-1.24 Å (-2.35 bohr). The oscillations of the wave function near the substrate have been truncated in order to show the oscillations through the outer Xe atoms better.

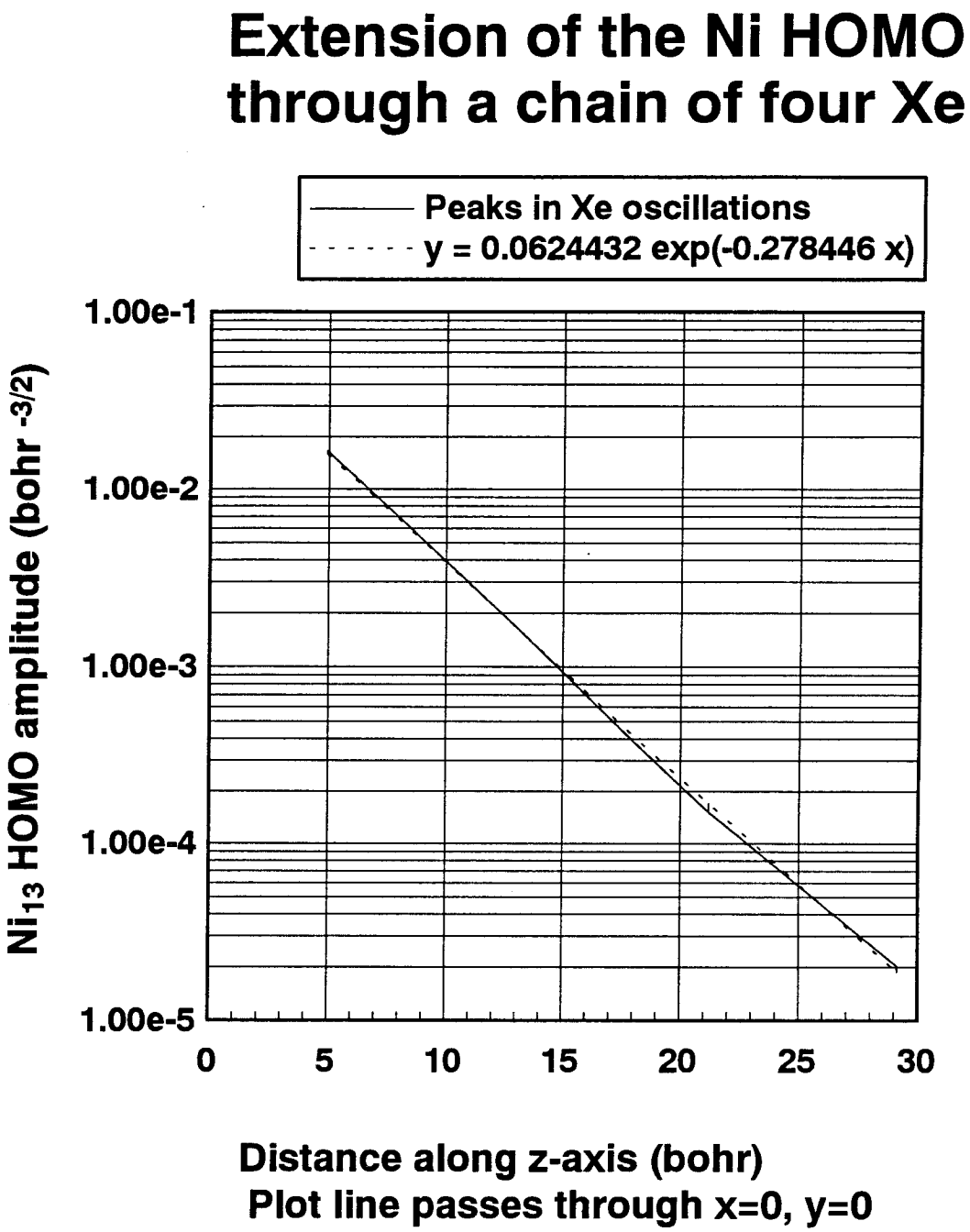


Figure 7:

Exponential fit to the peaks of the decaying oscillations in Figure 6.

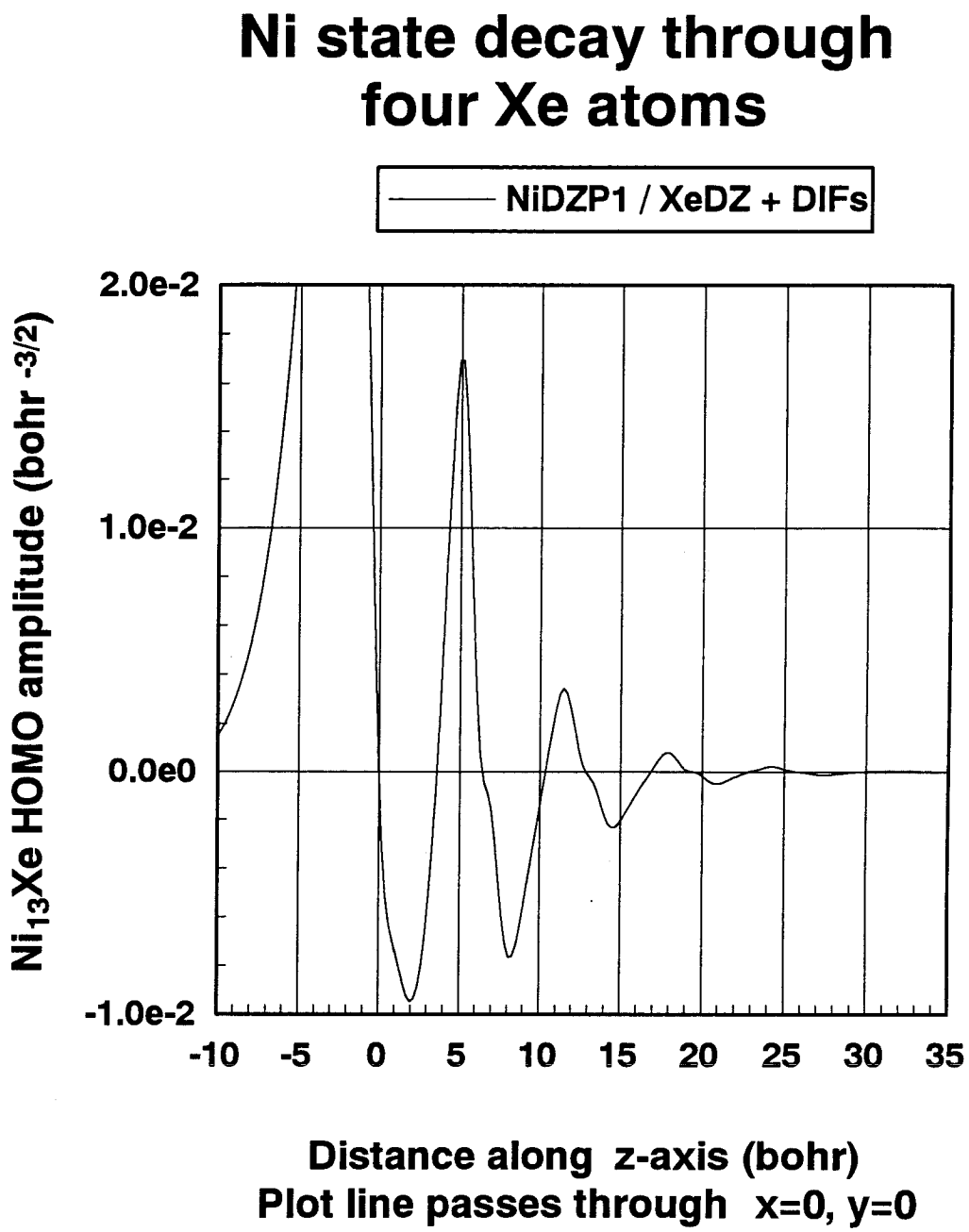


Figure 8:

Analogous to Figure 6 but with the Xe chain atoms located at 3.4, 6.8, 10.2, and 13.6 Å (6.4, 12.9, 19.3, and 25.7 bohr).

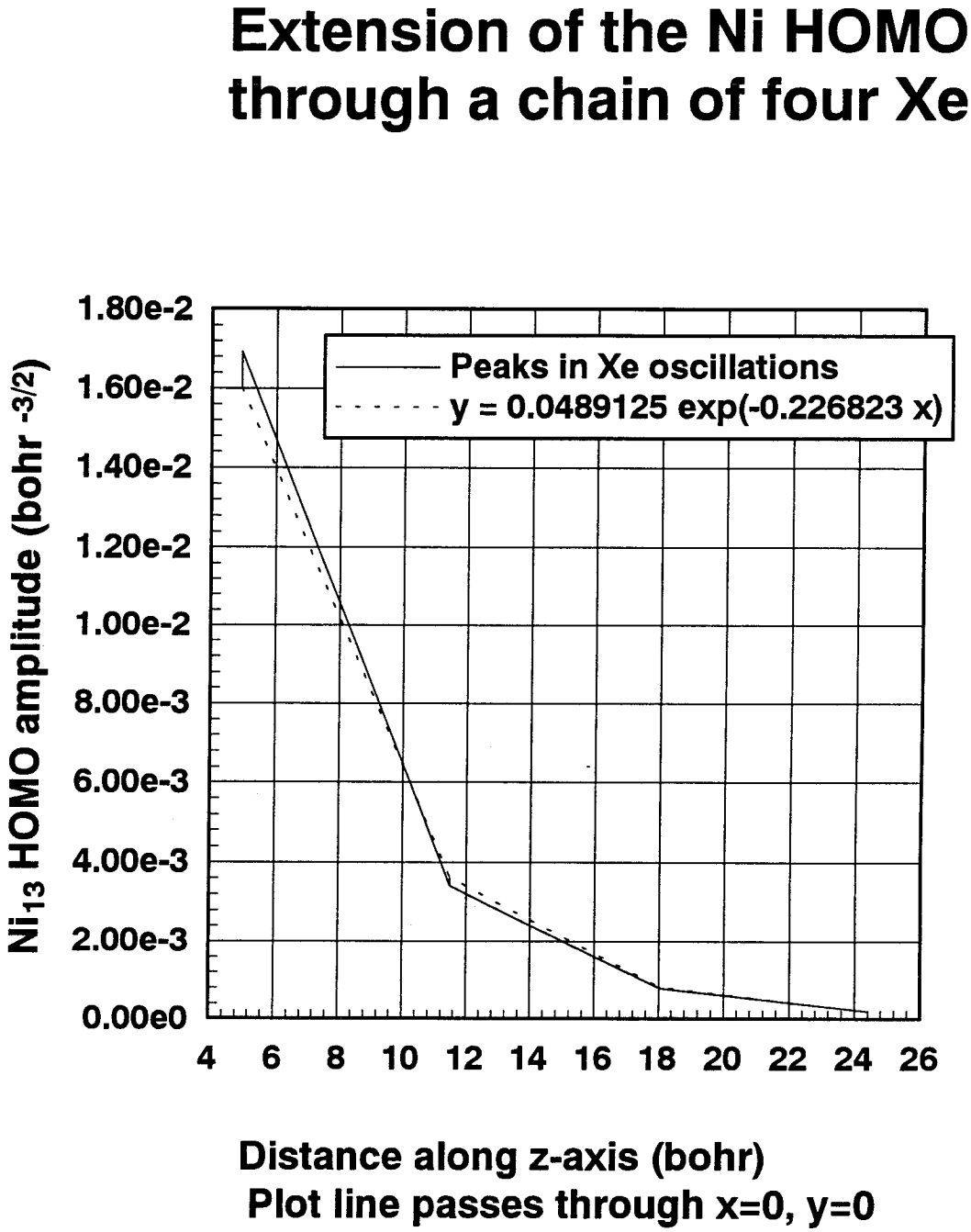


Figure 9:

Exponential fit to the peaks of the decaying oscillations in Figure 8.

3.2.5. Potential for Imaging Thick, Non-Bonded Adsorbates

SYNOPSIS:

Multiple layers of non-bonded adsorbates are unlikely to be imagable at low bias voltages (that sample only substrate states) unless the STM is stable in the picoamp range.

Our model can give some insight into what conditions must hold in order for the STM imaging of a weakly bound adsorbate to be successful. Figure 10 shows contours of constant density of the Ni HOMO near an adsorbed Xe atom. In a constant current experiment the tip will follow one of these contours, depending on the current set point. For example, if the current corresponds to a log-density of -6, the tip will scan over the Xe. Note, however, that if the current requirement is too high (e.g., log-density = -4) the tip will necessarily crash into the Xe and presumably simply move it along the surface instead of imaging it in place.

Eigler *et al.*, estimate their tip-sample distance at approximately 13 bohr. With respect to our model, this is in the range where the images should become possible without sweeping the Xe aside. Figures 11 and 12 show the same type of contour plot for a system with two stacked Xe atoms (z = 3.4 and 7.7 Å). In both cases, contours in the log-density range of -6 to -7 would predict a tip crash with the adsorbate.

The IBM Xe/Ni (110) experiment used a tip bias of +20 mV and a current set-point of 1 nA. If we assume these parameters correspond to imaging the -6 or -7 log-density contours in our model, then an STM that is tunneling at 10 pA might[17] be able to image systems with a few layers of non-bonded adsorbates via the substrate modulation mechanism. In general, however, we can conclude from these results that large adsorbates

---

[17]D. P. E. Smith, J. K. H. Hörber, G. Binnig, H. Nejoh, *Nature* **344** 641 (1990). They report collecting images at 50 pA current set points.

with more than a few non-bonded layers are unlikely to be imaged at low bias voltages which can only sample perturbed substrate states. Instead, sample states will need to become directly involved. Alternatively, the layer to layer interaction must be much greater than for the Xe-Xe case. Such alternatives could be explored using the techniques developed here.

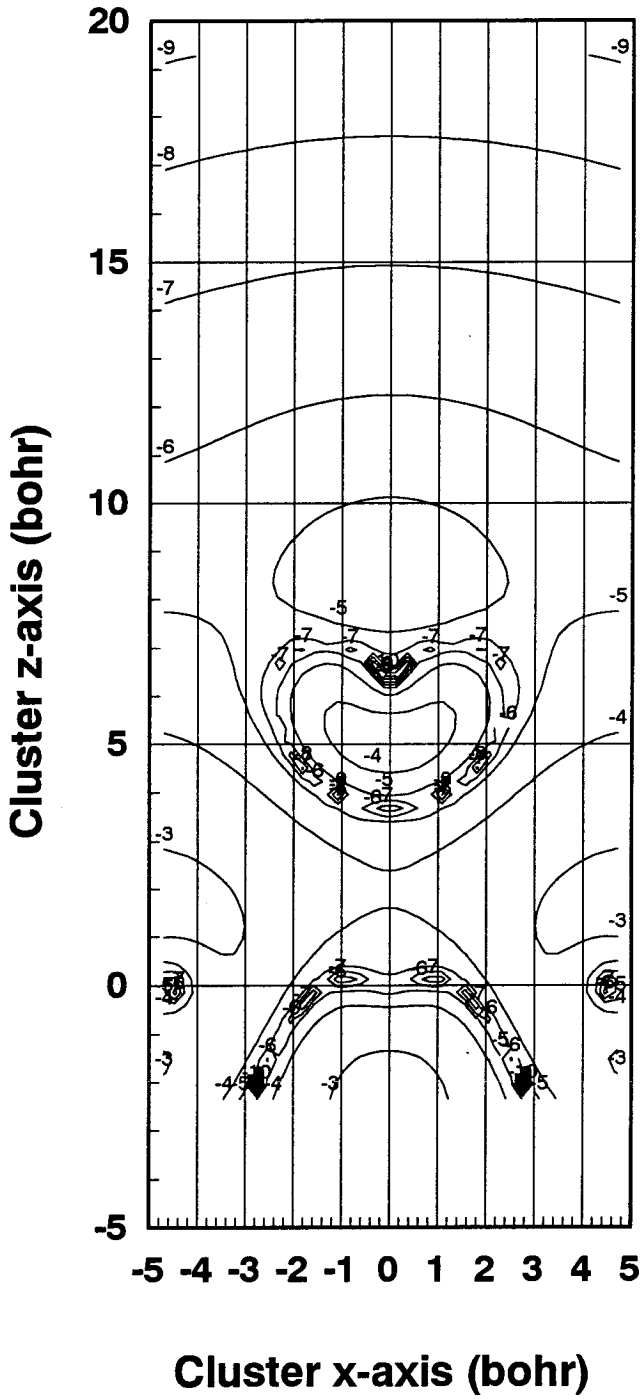


Figure 10:

The xz-plane is used as a surface on which contours of constant $\log_{10}\left(\left|\varphi(x, y=0, z)_{Ni_{HOMO}}\right|^2\right)$ are plotted. The Xe is positioned at $x = 0, y = 0, z = 3.4$ Å ($z = 6.425$ bohr).

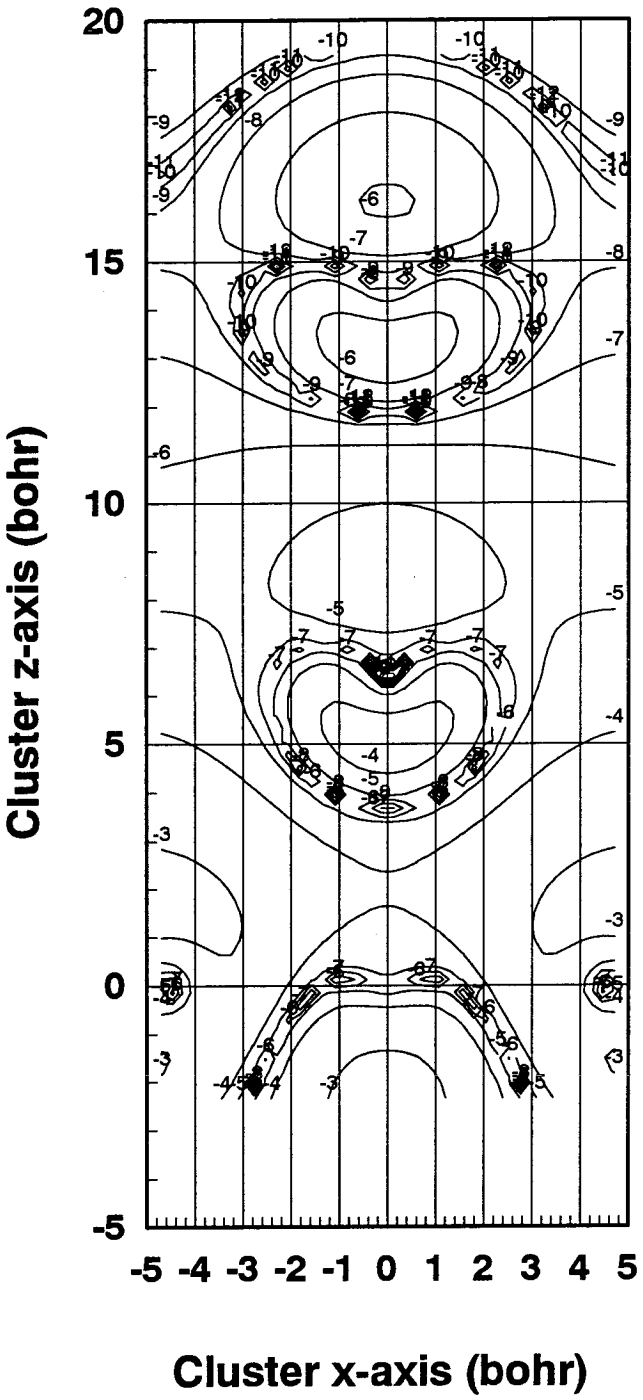


Figure 11:

Analogous to Figure 10 but with two Xe atoms, the first at 3.4 Å (6.425 bohr) and the second at 7.7 Å (14.551 bohr). These results suggest it would be difficult to image a dual layer of non-bonded adsorbates.

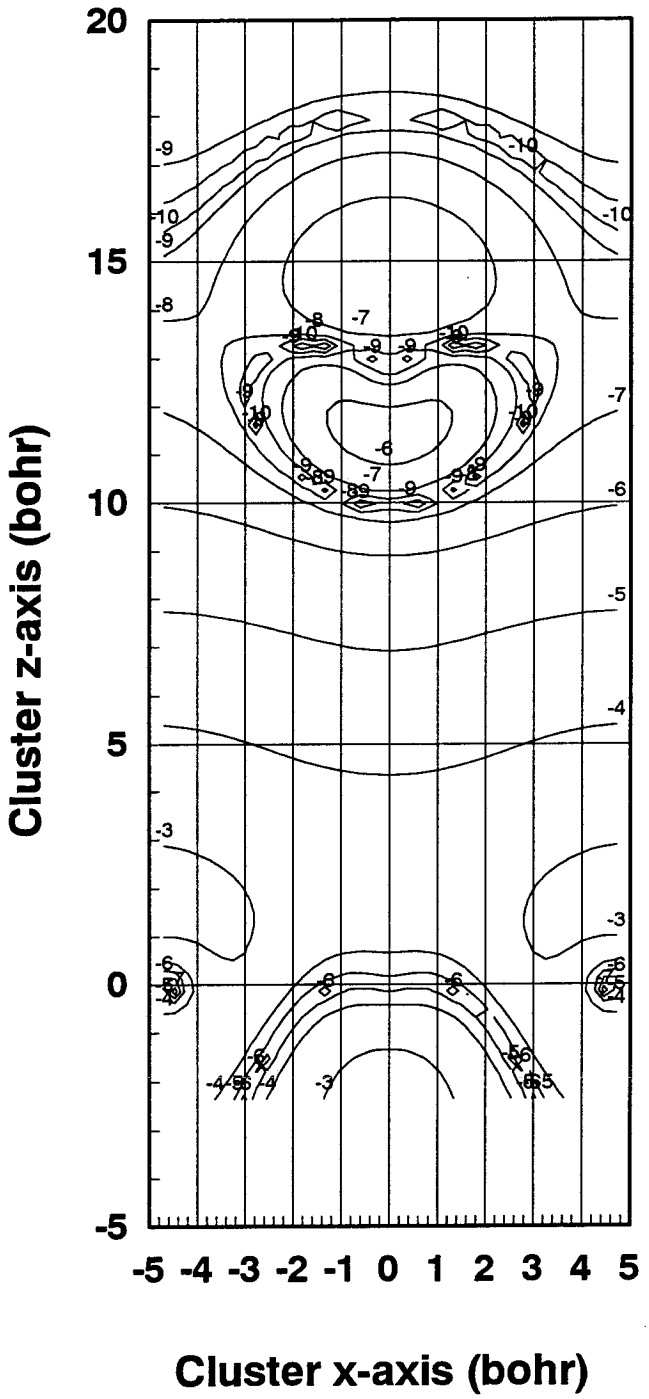


Figure 12:

Analogous to Figure 10 but with the single Xe atom moved to 7.7 Å (14.551 bohr). The results indicate that an adsorbate interacting this weakly with the substrate would be difficult to image.

3.2.6. Position of Xe Relative to the Ni Lattice

SYNOPSIS:

Larger model systems needed for verifying the Xe resting site on the Ni surface are discussed in light of the experience we have gained with smaller clusters.

Despite the remarkable successes in imaging adsorbates, STM experiments have not pinned down the relative position of the Xe on the Ni surface. The problem is analogous to that of the chemical identification problem in MoS_2: there is no unique feature to break the surface symmetry and thus allow an unambiguous assignment of features relative to the underlying nuclear centers. However, STM experimental and theoretical progress may soon make this assignment reliable. We describe why in the following paragraphs.

Although for most purposes one may think of the metal surfaces in STM metal-adsorbate systems as electronically flat, some experimentalists have obtained atomic resolution of Ni atoms.[12,18] Indeed, the Xe on Ni experiments show an underlying corrugation that is likely related to the Ni sites. However, any assignment for the location of Xe on the Ni (110) surface based on its registry with the small corrugations depends on proving how the "bumps" correspond to the underlying locations of Ni atoms. This proof is not so easy! Work by McAdon and Goddard,[19] for example, shows that Ni electrons probably localize in the octahedral and tetrahedral bulk interstitial sites. It is a reasonable speculation that the bumps in an STM image of Ni (110) might reflect the location of the underlying four-fold holes, or perhaps the Ni-Ni nearest neighbor bonding interactions.

---

[18]A. M. Baro, G. Binnig, H. Rohrer, Ch. Gerber, E. Stoll, A. Baratoff, F. Salvan, *Phys. Rev. Lett.* **52** 1304 (1984). Atomic resolution claimed for Ni(110).

[19]M. McAdon, Ph.D. thesis, California Institute of Technology.

Either way, the resulting image is commensurate with but displaced from the Ni (110) surface atomic positions.

With a properly scaled out basis set, larger cluster, and calculations that incorporate correlation (GVB might be sufficient), it would be a natural progression to extend our work to attempt a prediction of the small corrugations of the metal surface and to relate these to the perturbation from the Xe atom. Calculations for various sites of the Xe atom could then be correlated with STM experimental data to predict the correct resting site. What are the possible advantages of this approach over the usual approach of doing *ab initio* calculations of sufficient quality to predict relative binding energies? The primary advantage is simplicity. It would have to be determined whether it was cheaper to run a large STM model at the Hartree-Fock or GVB level, or a series of specially designed smaller clusters that can be run at the higher levels of correlation needed to get the non-bonding potential correct and consistent for various candidate binding sites.

A key criterion for a successful model indicated by our results is to build a cluster large enough to contain the unit of interest surrounded by the full chemical environment of the real system. In order to study Ni surface electronic density with respect to the surface atomic positions, we expect to need at least a unit cell of surface atoms surrounded by their nearest neighbor unit cells. For example, a 6x6 layer on top of a 5x5 layer would allow the study of Xe in the four-fold hollow site of the (110) surface. It also could be argued that the cluster should be deeper to avoid such a narrow quantum well in the direction perpendicular to the surface. By focusing on the density of the wave function near the center of the surface layer, this model might give results reliable enough to predict Ni atom surface corrugation. It would also be wise to compare the Hartree-Fock to the GVB results. Any discrepancies would indicate that correlation is important for determining the surface density and that localized states are playing an important role.

With advances in algorithms and computers,[20,60] large calculations like the ones discussed are just now becoming feasible for clusters of a size that does not require such careful attention to the cluster chemistry, i.e., less *finesse* is required to obtain reliable results.

3.2.7. A Two Xe System

SYNOPSIS:

A model with two adjacent Xe atoms on Ni (110) is studied. The Ni HOMO and nature of the Xe perturbation are found to be consistent with the single Xe case.

To explore the Xe/Ni (110) experiment a little further, we calculated the 131 basis function case of Xe_2/Ni_{23}. Here, the Xe basis set was again Xe DZ + DIFs and the Ni basis set was Ni DZP1. The top layer of Ni has been extended to 15 atoms, 5 atoms in the <110> direction and 3 atoms in the <001> direction. The subsurface was augmented from 4 atoms to 8 atoms, a 4x2 layer under the rectangular hollows of the surface. The Xe atoms were placed at 3.25 Å above the Ni surface, slightly closer than in the previous calculations. The Xe atoms rest above the only two surface Ni atoms with a full complement of surface nearest neighbors.

Figure 13 shows that the orbital modeling the bulk Ni states at the Fermi energy is identical in nodal structure to the Ni_{13} case. Also, the type of perturbation induced by each Xe atom is identical to that of the single atom case. This gives us confidence that the smaller cluster is correctly modeling the important features of the system.

---

[20]In addition to ever faster supercomputers, computers like the Kendall Square Research shared memory parallel processo are available for under $1,000,000.

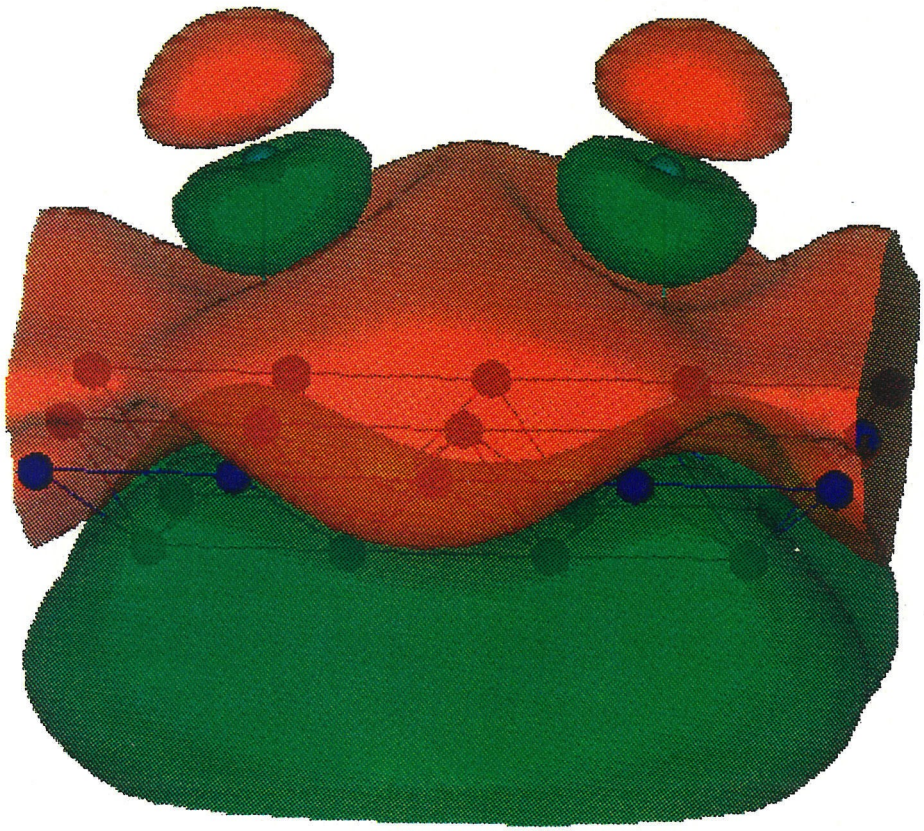


Figure 13:

Ni_{23} HOMO with two Xe atoms present. Isosurface is +/- 0.003 $bohr^{-3/2}$.

3.3. Cyclopropene on Ni (110)

SYNOPSIS:

We conclude that it is unlikely that the STM operating at low bias voltages could distinguish single from double carbon-carbon bonds in small molecules physisorbed to a Ni surface. Instead, it is more likely that any spatially resolved features of a physisorbed organic molecule would reflect points of underlying closest contact with the metal surface.

3.3.1. Motivation

We would like to see if the imaging mechanism at work in the Xe/Ni (110) case could be used to distinguish chemically distinct sites in molecular adsorbates. Therefore we have created the simplest model system with two distinct types of carbon-carbon bond: cyclopropene on Ni (110). As in the Xe/Ni (110) case, we expect that cyclopropene will have states too deep and too high with respect to the Ni Fermi level to be imaged at low bias voltages. Therefore, we will be examining perturbations to the same state of the Ni_{13} cluster we used in the Xe/Ni (110) calculations, but the perturbations will be induced by the organic molecule.

The model system we will use is fictitious in the sense that this species is probably not chemically stable on the Ni surface. Also, the distance to the surface was not optimized. We learned from the Xe case, however, that changing the distance primarily affects the magnitude of the perturbation, as long as the non-bonded states are not compressed unreasonably. Therefore, we will use this system to answer one question: is it likely that imaging a physisorbed organic at low bias voltage could resolve the single from double carbon-carbon bonds?

3.3.2. Cyclopropene Perturbations of the Ni HOMO

Figures 14 to 19 present a series of views of $\left|\varphi(x, y, z = 4.5\mathring{A})_{Ni_{HOMO}}\right|^2$ as we move the C_3H_4 from the left to the right over the Ni_{13} cluster. These images approximate what should result from STM image taken under conditions similar to the Xe/Ni experiment. However, by comparing Figures 14 and 19 we see immediately that our model is too small to be reliable for all possible orientations of the C_3H_4 with respect to an underlying Ni atom. The two sites in Figures 14 and 19 are identical on the bulk Ni (110) surface. Therefore, the C_3H_4 must generate identical perturbations to the surface. Yet, we see a shift in the center of the perturbation with respect to the cyclopropene molecule. Therefore, the exact position of the perturbation in these two extreme cases is due to the edge effects of the Ni cluster.

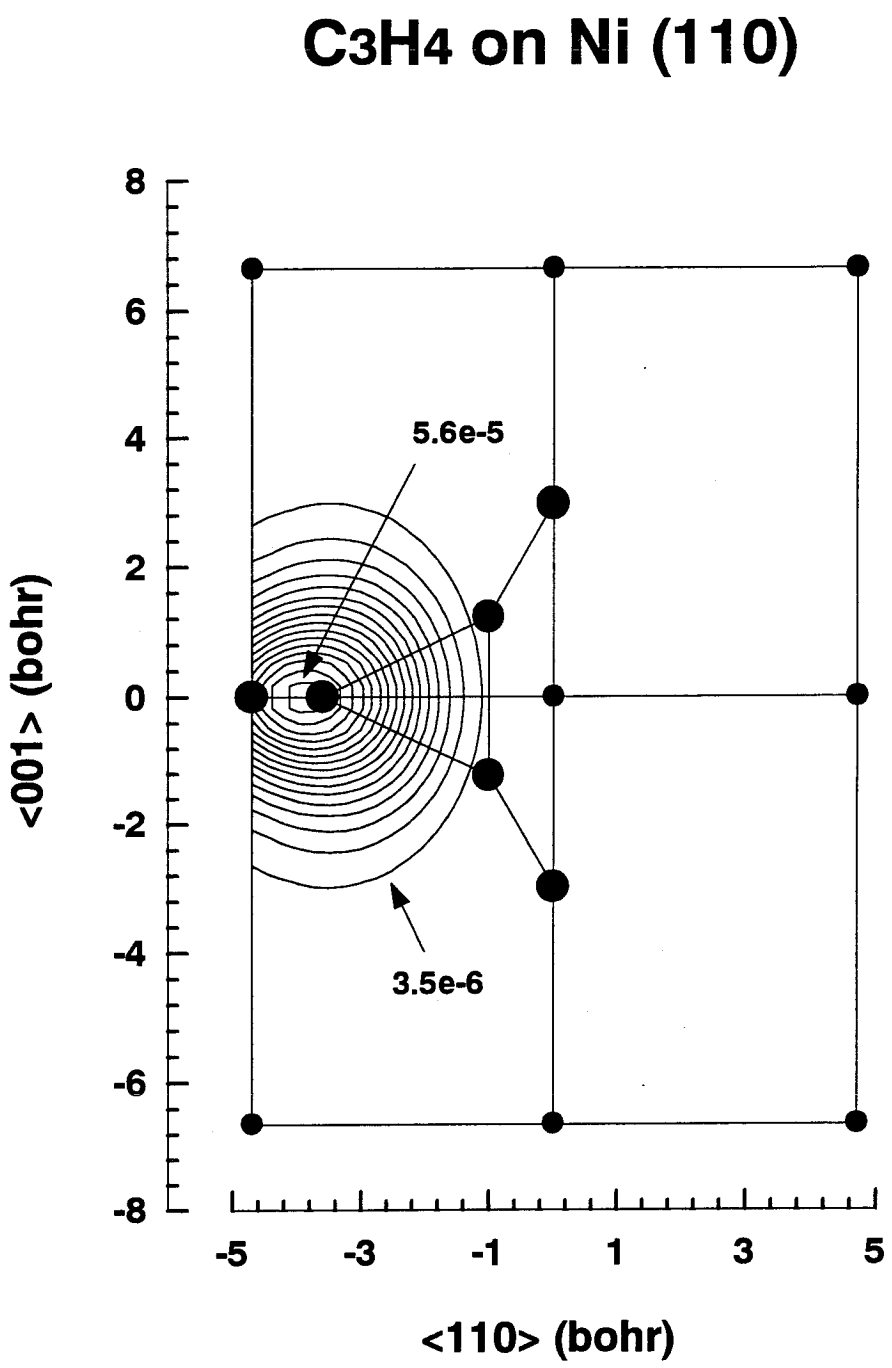


Figure 14: Small circles show the positions of the nine surface Ni atoms. Large circles are positions of the C_3H_4 atoms projected onto the surface. The contours are constant density of the Ni_{13} HOMO (bohr^{-3}). Contours start at and are in increments of 3.5e-6 bohr^{-3}. The single bonded carbon is at x = -3.60 bohr, y = 0, z = 5.67 bohr (3.0 Å).

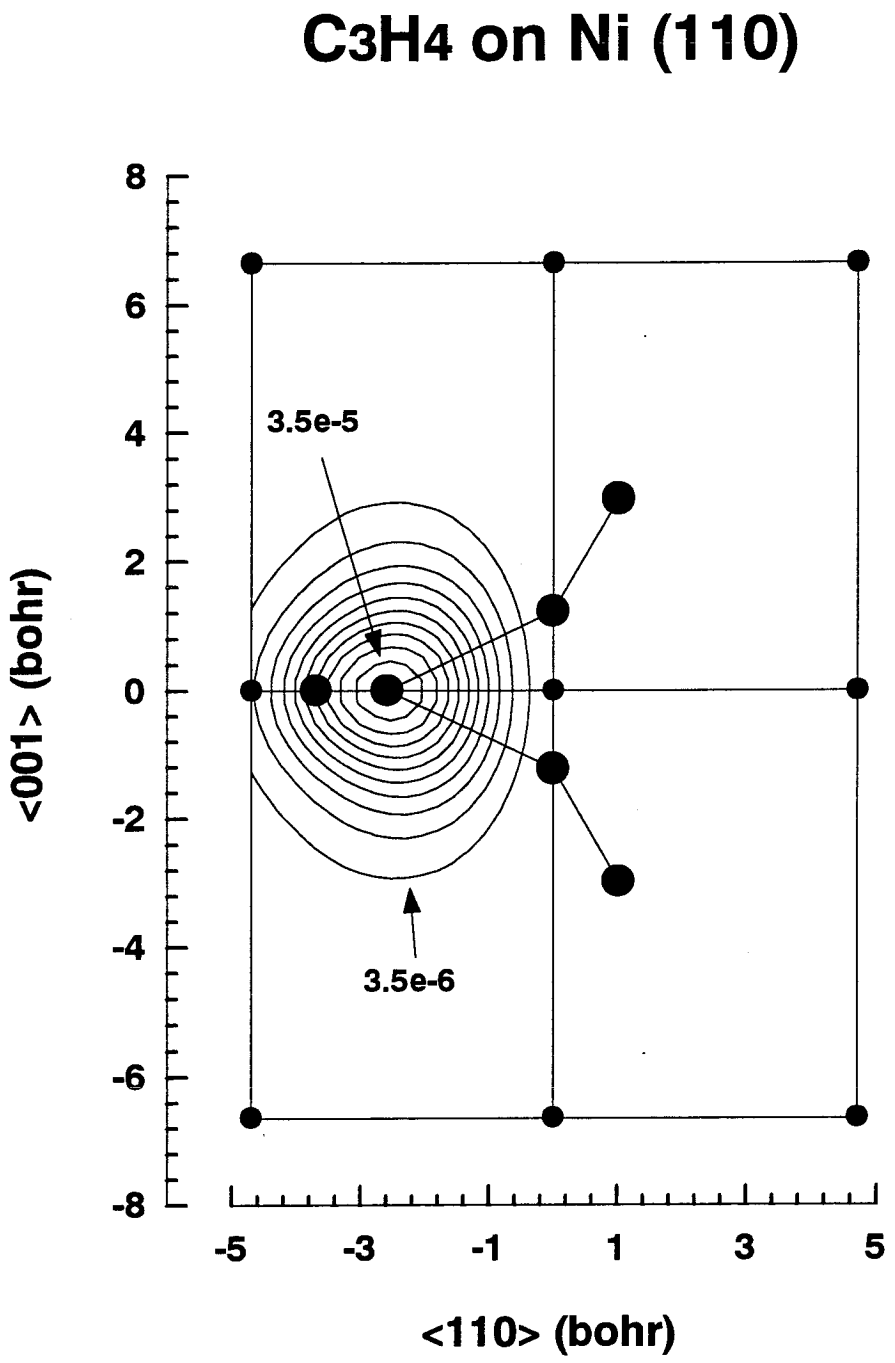


Figure 15:
Analogous to Figure 14 but with the single bonded carbon at x = -2.59 bohr.

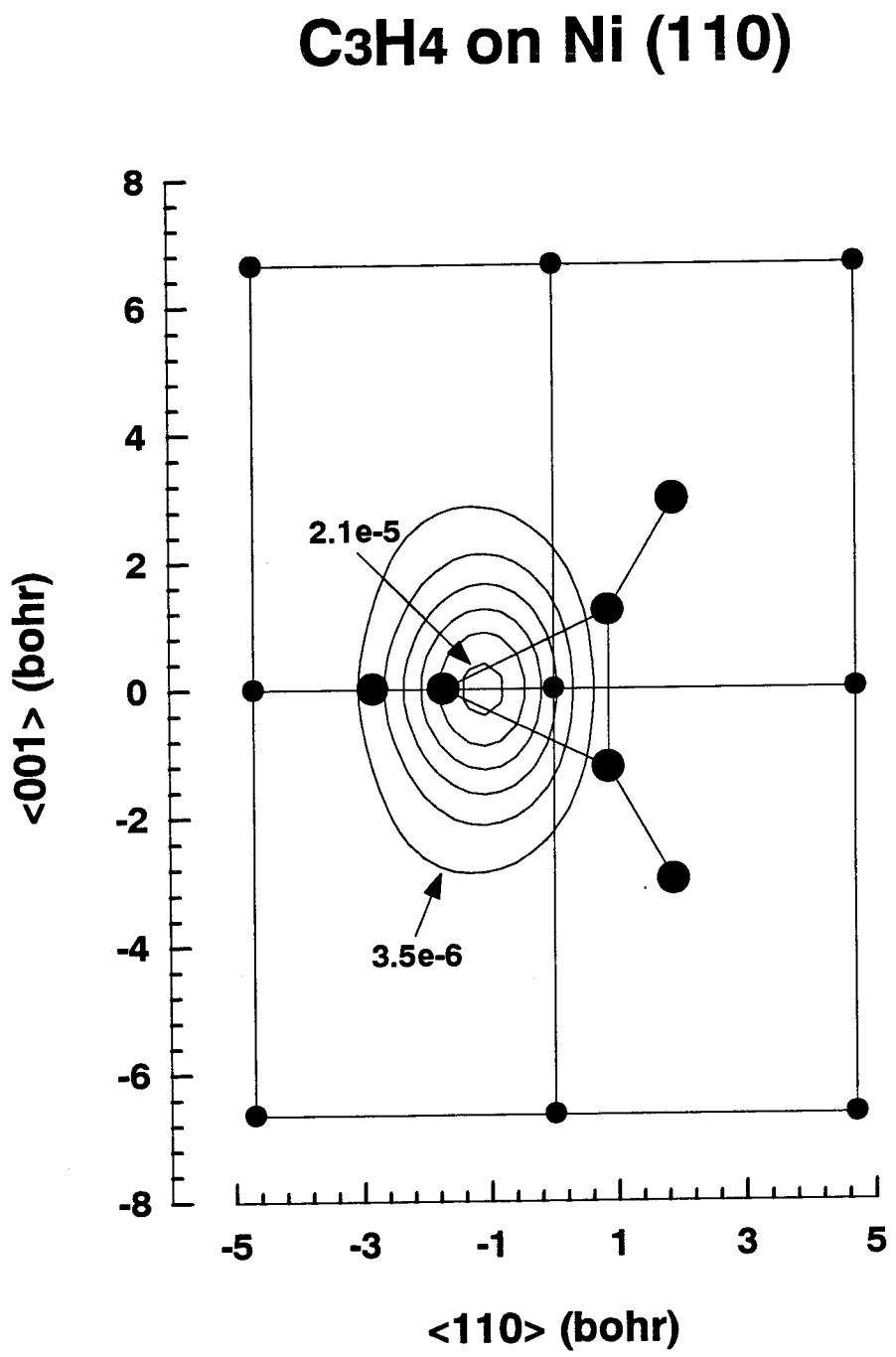


Figure 16:
Analogous to Figure 14 but with the single bonded carbon at x = -1.72 bohr.

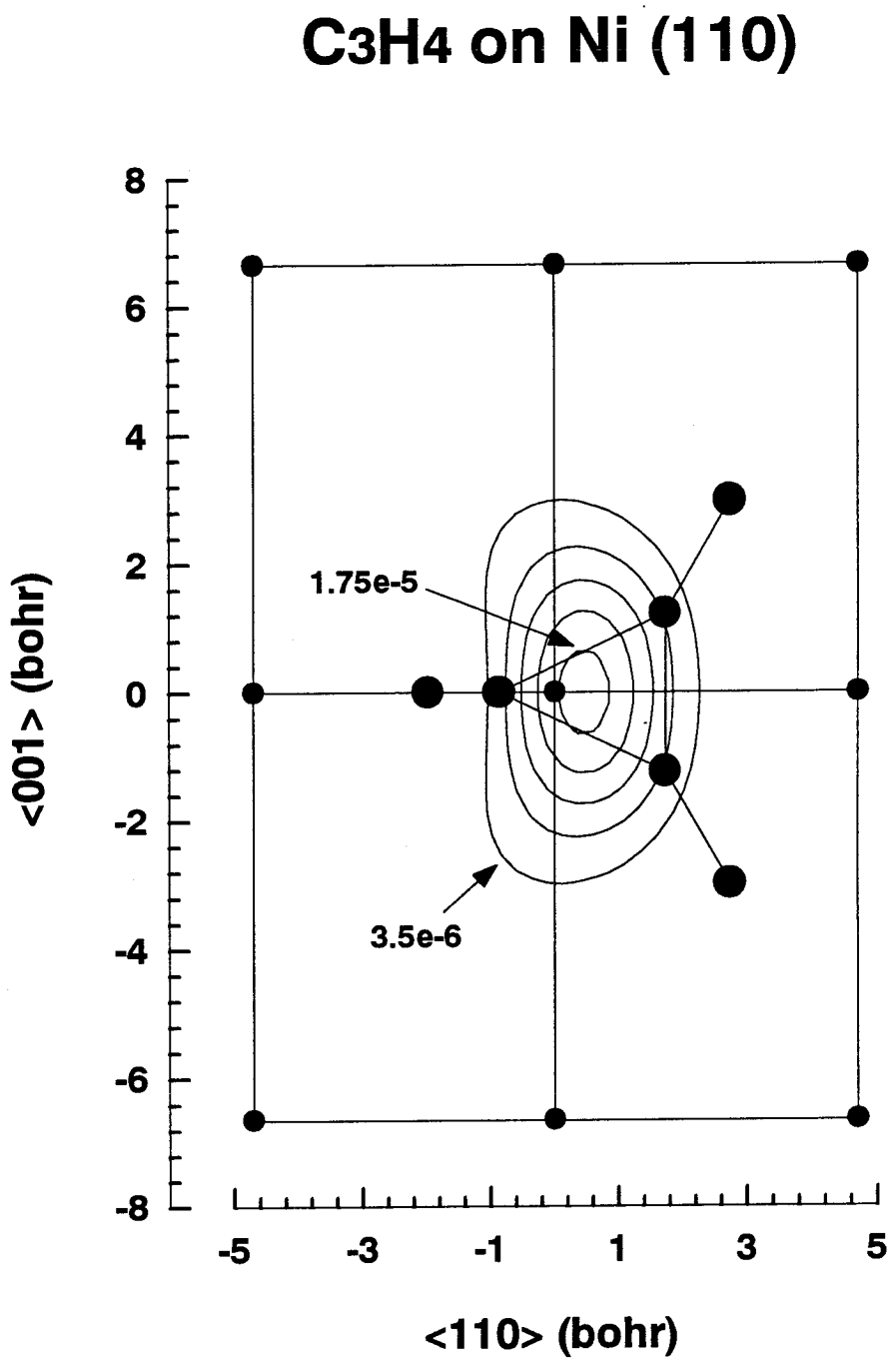


Figure 17:

Analogous to Figure 14 but with the single bonded carbon at x = -0.86 bohr.

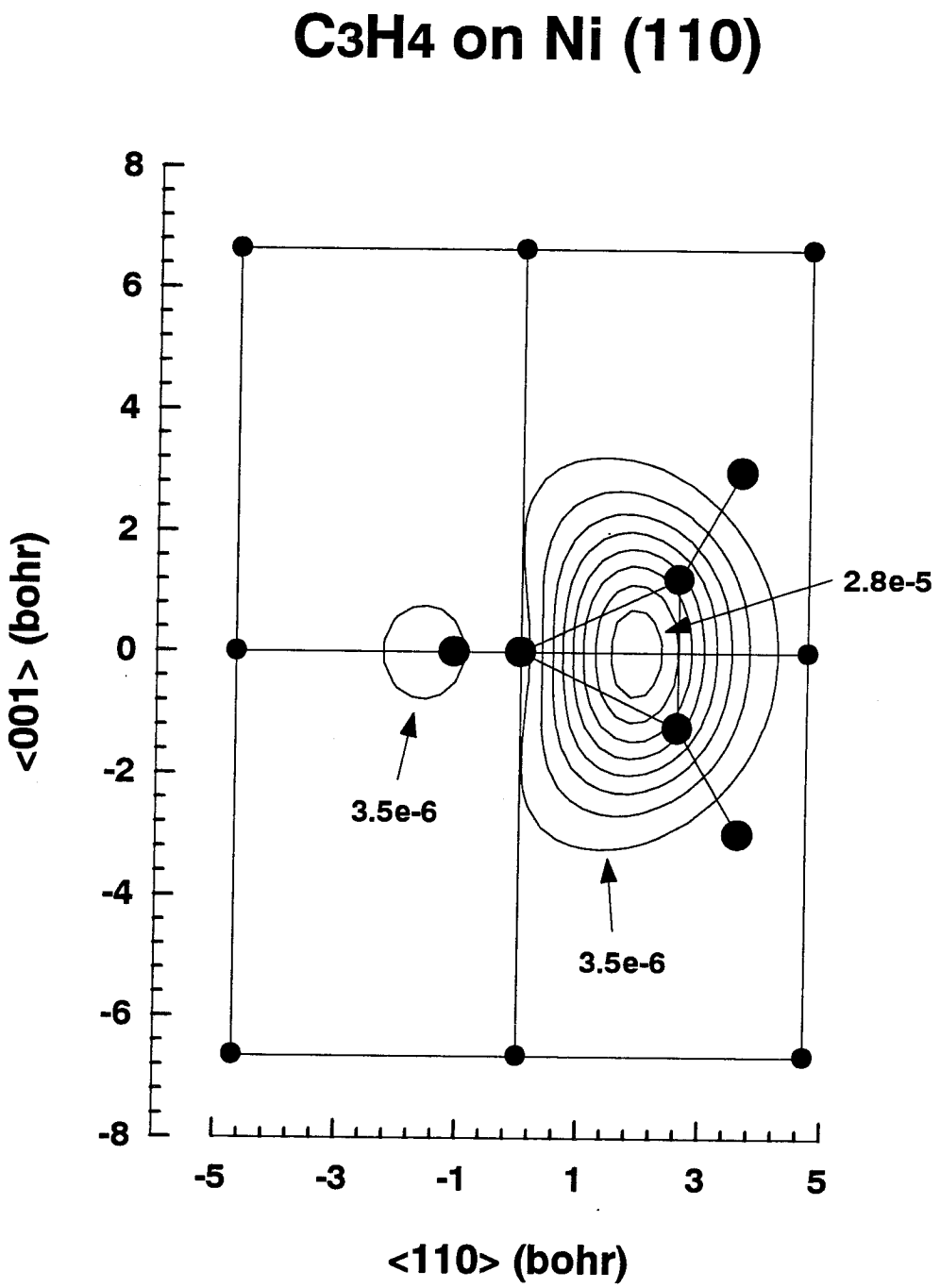


Figure 18:
Analogous to Figure 14 but with the single bonded carbon at x = 0.00 bohr.

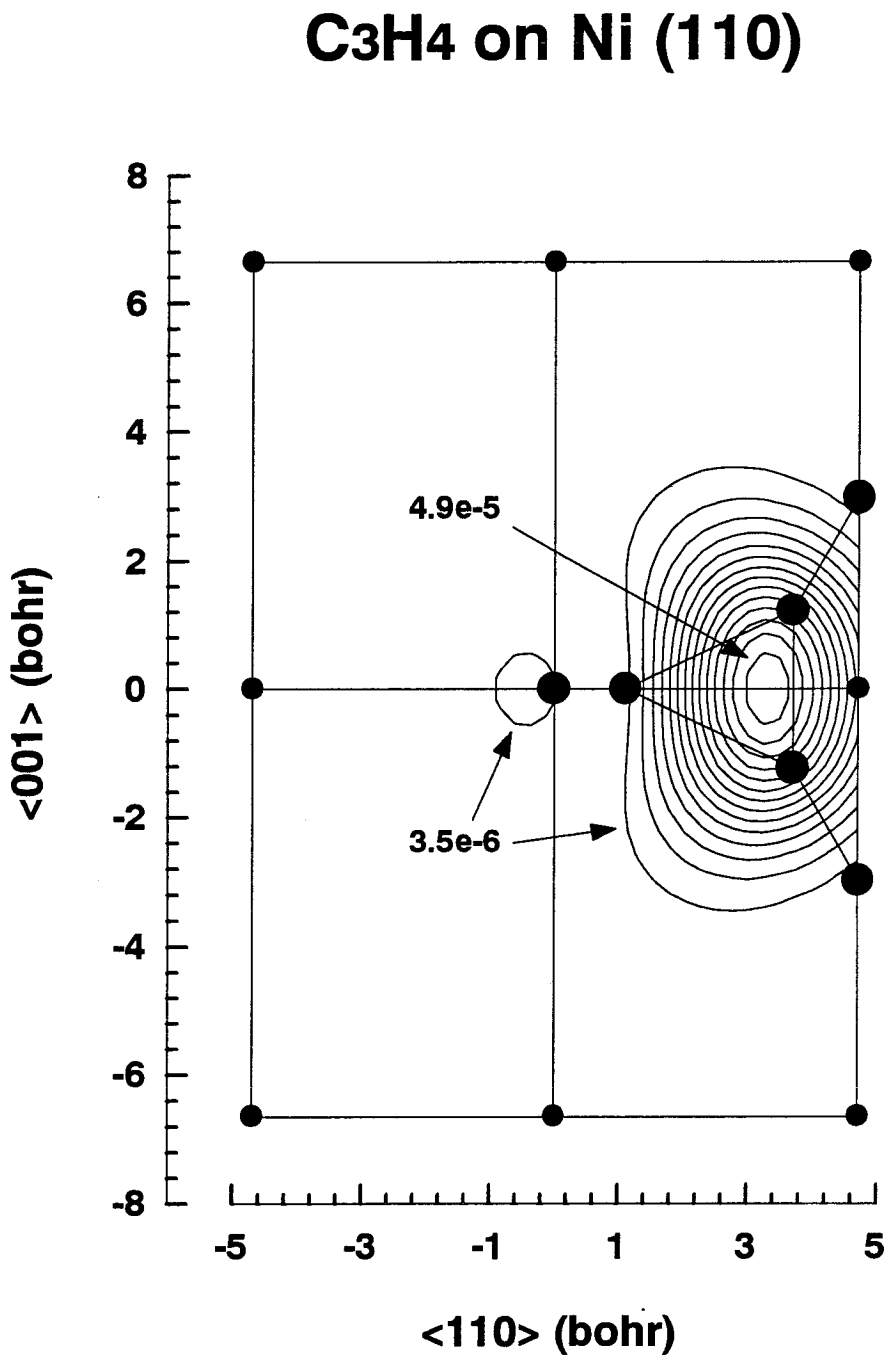


Figure 19:
Analogous to Figure 14 but with the single bonded carbon at x = 1.11 bohr.

These results are not surprising since the Ni cluster was designed to have a good description of the chemical environment surrounding a single atom, not a complete molecule. Having taken these two-dimensional cuts of HOMO density, we might decide there is nothing further to learn from this cluster model. However, Figures 20 to 24 show full three-dimensional views of the Ni HOMO and suggest several conclusions about this system not apparent from the two-dimensional plots.

The first thing to note is that the interaction with the Ni HOMO is qualitatively similar to the Xe/Ni case: the perturbation appears as a *p*-type oscillation in the wave function near the adsorbate. This simple behavior is exhibited in spite of the more complex nature of the adsorbate.

The second thing we note is that the exact shape and maximum position of the perturbation is not correlated with the presence of the double bond in C_3H_4; rather it depends on C_3H_4 overlapping with either of the two broad maxima in the Ni HOMO above the surface. The shape of the Ni HOMO is not a perfect plane wave. Even in the bare Ni_{13} cluster the HOMO shows a slight depression in the center (see Figure 2). This is due to the finite size of the cluster. Nevertheless, the three-dimensional views show that the perturbation is greatest where the adsorbate is closest to the Ni HOMO. If the Ni surface did present a perfect plane wave, then we would expect the maximum perturbation to appear shifted toward the single bonded carbon where the hydrogens are closest to the surface.

Because the state being imaged is not a state of the adsorbate, there is very little energy penalty for moving the perturbation maximum relative the adsorbate since only the tail of the Ni wave function is being affected. (The Hartree-Fock energy with the adsorbate at either end of the cluster is about 3.5 kcal/mol lower than with it at the center.) We do not expect the STM to be able to distinguish the double bond in this case,

especially when we consider the blurring effect of a finite tip size (see next section). In other systems we will explore next, we will see that if states of the adsorbate are directly contributing to the tunneling current, the location of the current maximum is much more definite.

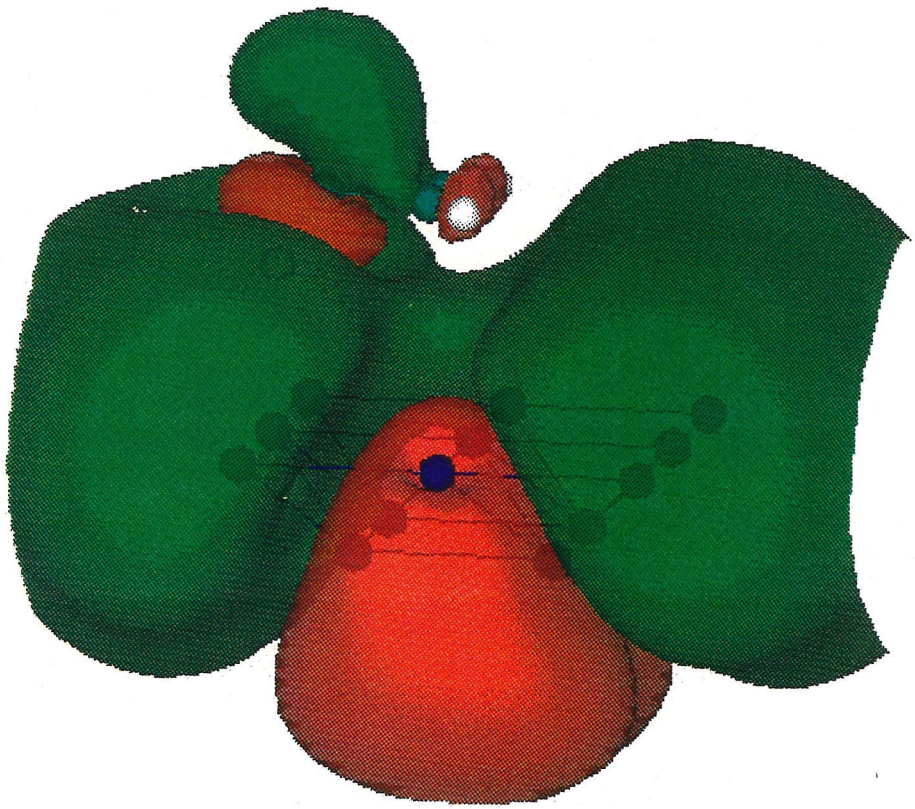


Figure 20:
$\varphi_{Ni_{HOMO}}$ plotted at +/- 0.003 bohr$^{-3/2}$.
The single bonded carbon is at x = -3.60 bohr (-1.91 Å).

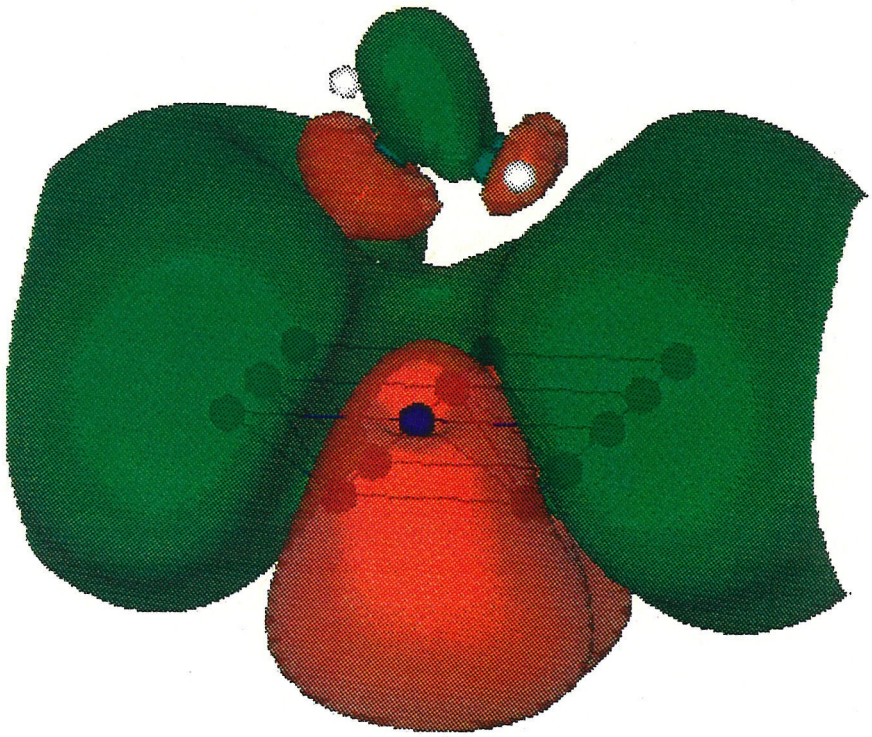


Figure 21:
$\varphi_{Ni_{HOMO}}$ plotted at +/- 0.003 bohr$^{-3/2}$.
The single bonded carbon is at x = -1.72 bohr (-0.91 Å).

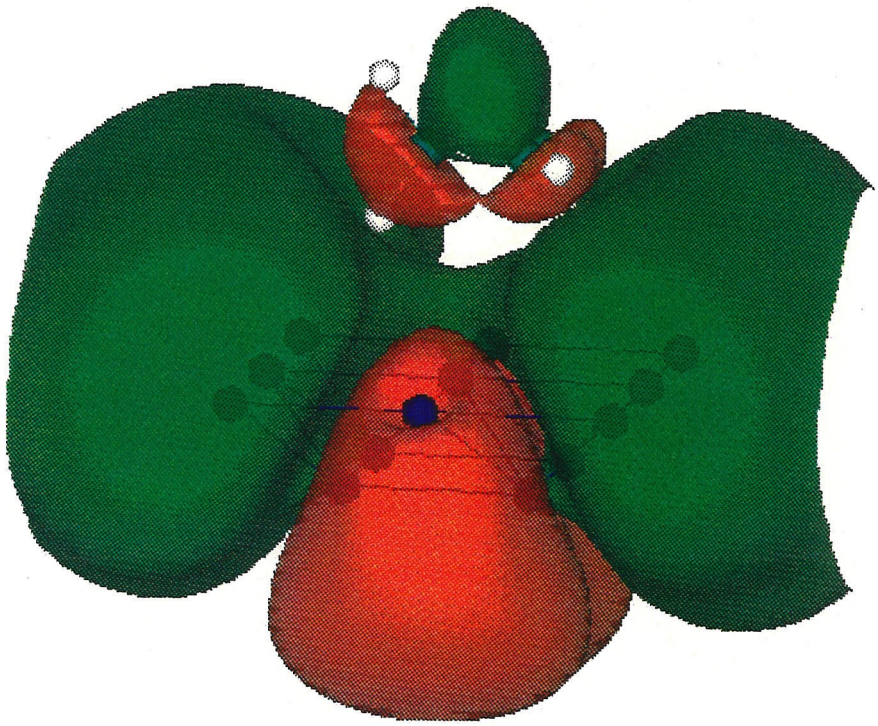


Figure 22:
$\varphi_{Ni_{HOMO}}$ plotted at +/- 0.003 bohr$^{-3/2}$.
The single bonded carbon is at x = -0.86 bohr (-0.46 Å).

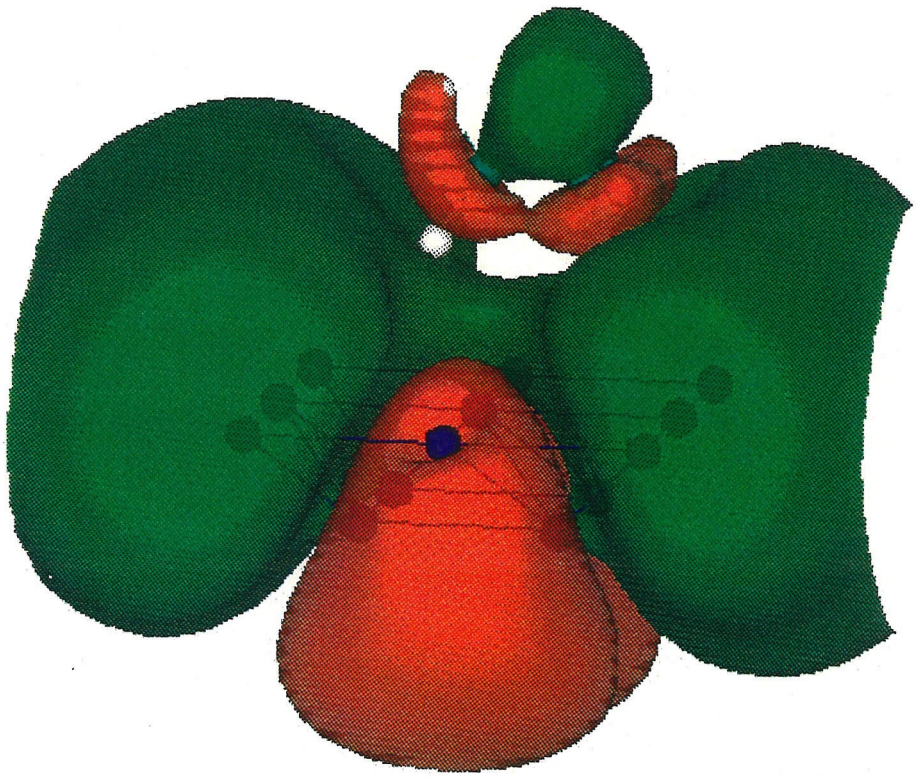


Figure 23:
$\varphi_{Ni_{HOMO}}$ plotted at +/- 0.003 bohr$^{-3/2}$.
The single bonded carbon is at x = 0.00 bohr.

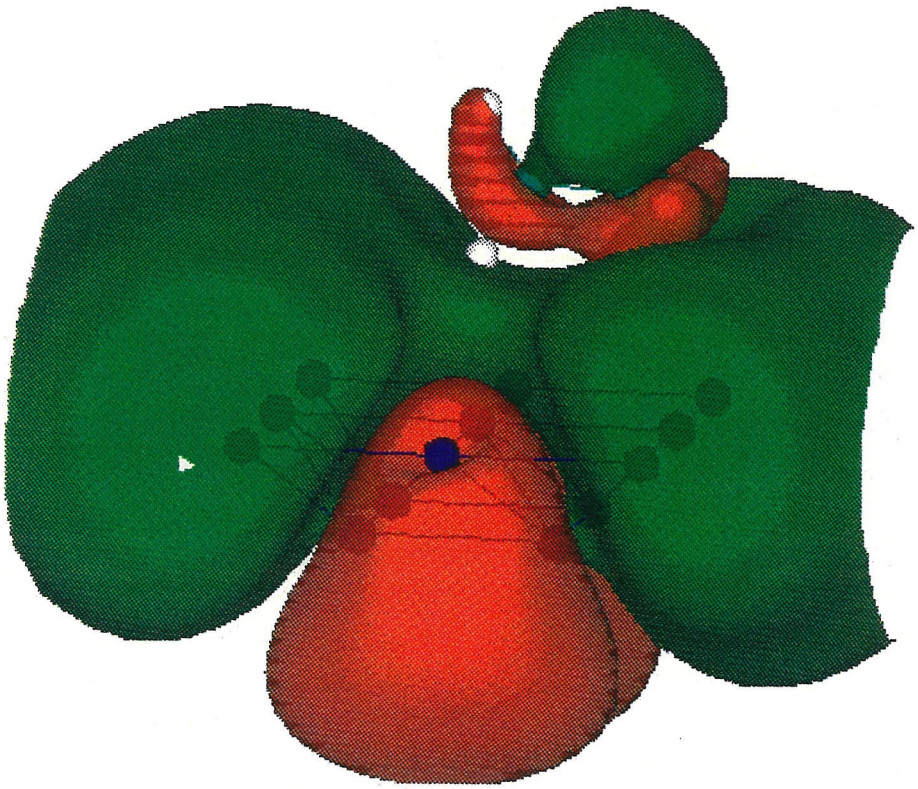


Figure 24:
$\varphi_{Ni_{HOMO}}$ plotted at +/- 0.003 bohr$^{-3/2}$.
The single bonded carbon is at x = 1.11 bohr (0.59 Å).

3.4. Appendix

3.4.1. Cluster Coordinates

The coordinates for the Ni_{13} cluster are listed in Table 1.

Atom Label	Basis Set	X (Å)	Y (Å)	Z (Å)
N000	Ni DZP1	0.0	0.0	0.0
N100	Ni DZP1	2.487	0.0	0.0
N900	Ni DZP1	-2.487	0.0	0.0
N010	Ni DZP1	0.0	3.5172	0.0
N110	Ni DZP1	2.487	3.5172	0.0
N910	Ni DZP1	-2.487	3.5172	0.0
N090	Ni DZP1	0.0	-3.5172	0.0
N190	Ni DZP1	2.487	-3.5172	0.0
N990	Ni DZP1	-2.487	-3.5172	0.0
NS11	Ni DZP1	1.2435	1.7586	-1.2435
NS91	Ni DZP1	-1.2435	1.7586	-1.2435
NS19	Ni DZP1	1.2435	-1.7586	-1.2435
NS99	Ni DZP1	-1.2435	-1.7586	-1.2435

Table 1

3.4.2. Positioning of the Xe Above Ni (110)

For the purposes of our model of Xe imaging on Ni, the exact placement of the Xe is not critical. The following factors went into our choice of the 3.4 Å on-top site. One way of estimating the closest contact is to average the atomic radii:

$$\text{Ni-Ni} = 2.487\text{Å}$$

$$\text{Xe-Xe} \approx 4.36\text{Å}^{21}$$

$$\text{Ni-Xe} \approx 1.2435 + 2.18 = 3.42\text{Å}$$

---

[21]W. A. Goddard, III, Chemistry 120 Notes, California Institute of Technology

Most of the Ni-adsorbate work was carried out in January of 1990. We located no experimental data from 1987 to 1990 that contained data for Xe on Ni relating to the adsorption site or interatomic distances. One paper we located, however, did contain Xe on Pt (111) data.[22] This source gave an Xe-Pt distance of 3.1 Å with a note added in the proof that gave a distance of 3.49 Å. We decided the 3.4 Å distance based on averaging radii was suitable for our purpose.

Bethune *et al*, suggested a 3-fold hollow site for Xe on Pt (111). Since the physisorbed Xe interacts with the Ni surface only through van der Waals interactions, the most favorable location will depend sensitively on the ability of the Xe and the Ni surface electrons to polarize. On the Ni (110) surface, an on-top site, four-fold hollow, or bridging site would not be distinguished energetically at the level of theory (Hartree-Fock) we employed to study the Xe interaction with the Ni tail. To the extent that the Ni wave function behaves as a plane wave above the surface, this will not affect our conclusions. We chose the on-top site because it allows the most economical calculation that allows Xe to interact with a Ni atom that has a full chemical environment of surrounding Ni atoms. If we had chosen a two-fold bridge site, then we would need to worry that the two Ni atoms involved in the bridge were surrounded with a full chemical environment of Ni atoms. The two-fold bridge site requires more atoms in the cluster and hampers the ability to study other features of the system in a reasonable amount of time (e.g., basis set scaling). The same is true for the four-fold hollow site.

Subsequent to these calculations, two papers appeared which support the idea that Xe is actually adsorbed in on-top sites of Pt (111). The first study was by Gottlieb[23] who

---

[22]D. S. Bethune, J. A. Barker, C. T. Rettner, *J. Chem. Phys.*, **92** 6847 (1990).

[23]J. M. Gottlieb, *Phys. Rev. B* **42** 5377 (1990).

discovered a signature for the on-top site in available diffraction data and also did a theoretical study. Shortly thereafter, Müller[24] presented a large basis set density functional calculation of the Xe-Pt(111) system. Two conclusions of this work are important here. First, the preferred site is the on-top site with a binding energy of 307 meV (experimental is 277 meV). The preference for the on-top site was explained by the polarization of the metal by partial occupation of empty Pt 5*d* states via transfer from each Xe of approximately 0.110*e*. Since we specifically exclude Ni *d*-functions from our calculation, it is not unexpected that we do not see the shallow potential well that leads to the on-top binding site.

The second conclusion of interest is the Xe-Pt bond distance of 3.0 Å. This distance is shortest at the on-top site and maximizes the favorable interaction with between Xe and Pt 5*d* states. For our calculations, the implication of a shorter Ni-Xe distance is an increase in the magnitude of the perturbations to the Ni surface function that we have seen.

3.4.3. Coordinates of Cyclopropene

The coordinates for cyclopropene[25] are listed in Table 2.

---

[24]J. E. Müller, *Phys. Rev. Lett.* **65** 3021 (1990).

[25]"Structure Data of Free Polyatomic Molecules", Vol 7, Ed. K.-H. Hellwege, Springer-Verlag (1976), page 228.

Atom Label	Basis Set	X (Å)	Y (Å)	Z (Å)
C1	C1	0.00000	0.00000	3.00000
C2	C1	1.36848	0.65000	3.00000
C3	C1	1.36848	-0.65000	3.00000
H11	HS2Z4	-0.58644	0.00000	3.91523
H12	HS2Z4	-0.58644	0.00000	2.08477
H2	HS2Z4	1.90509	1.57571	3.00000
H3	HS2Z4	1.90509	-1.57571	3.00000

Table 2

3.4.4. Basis Sets

The origin of the Ni basis sets used here may be found in Upton's thesis.[26] The standard Xe basis set comes from Hay and Wadt (Los Alamos).[27] We reproduce those exponents here and present exponents for the extended functions used in this work so that the plotted results may easily be correlated with the underlying basis set. The various combinations used are summarized and named in a final table.

The effective core potential used in all cases for Ni is from Upton, Melius, and Goddard[28]. It was designed to allow treatment of the *s*-band of Ni independently of the *d*-band which was found to be reasonably well separated spatially and energetically. Thus, the valence *d*-electrons are treated in the effective core potential.

---

[26]T. H. Upton, Ph.D. thesis, California Institute of Technology (1980).

[27]P. J. Hay, W. R. Wadt, *J. Chem. Phys.* **82** 270 (1985).

[28]C. Melius, Ph.D. thesis, California Institute of Technology (1973).

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
S/1	2.394170	-0.012851
S/1	0.918169	-0.151781
S/1	0.130176	0.410190
S/2	0.046392	1.0
P/1	0.14	1.0

Table 3: Ni 2*s*1*p* (DZP1) basis set.

The basis of Table 3 is referred to as the Ni DZP1 basis set (valence double-zeta plus polarization).[29] This is the original *s*-function 1-electron basis set for Ni combined with a single polarization function.

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
P/1	2.035040	-0.0259
P/1	0.780444	0.0385
P/1	0.110650	0.6211
P/2	0.039433	1.0

Table 4: Ni P2 (double zeta polarization) function.

The polarization functions of Table 4 are the original double zeta polarization function developed by Upton and Goddard.[26] This set will occasionally be used to replace the single zeta polarization function of Table 3 and be referred to as Ni DZP2.

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
S/1	0.0165331	1.0

Table 5: Ni diffuse *s*-function (Ni DIFs).

---

[29]The *p*-function exponent was optimized by Jason K. Perry for a Ni_{14} cluster (three layers, 5-4-5) at the Hartree-Fock level.

This diffuse *s*-function was derived in the usual way by scaling out the ratio of the previous two *s*-functions in the Ni DZP1 basis set using the ratio

$$\frac{0.130176}{0.046392} = \frac{0.046392}{\mathbf{0.0165331}}.$$

We use this exponent to augment the other Ni basis sets to achieve proper exponential decay where necessary.

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
S/1	0.7646	-1.514366
S/1	0.5322	1.727028
S/2	0.1491	0.633809
P/1	1.211	-0.140522
P/1	0.3808	0.621298
P/2	0.1259	0.536626

Table 6: Xe 2*s*2*p* (DZ) basis for Xe.

Table 6 is the standard valence double zeta basis set for Xe developed by Hay and Wadt.[27]

The following diffuse *s*-function and diffuse *d*-functions for Xe were optimized for excited states of Xe at the Hartree-Fock level. These functions were used to explore the propensity of the Ni wave function to use excited states of the Xe.

Function type	Radial exponent α_i	Configuration optimized	Hartree-Fock energy
S	0.022	$s^2p^4p_z^1s^{*1}$	-14.919409
D	0.015	$s^2p^4p_z^1d_{xy}^1$	-14.845712

Table 7: Xe diffuse functions (DIFs and DIFd).

3.4.4.1. Proper Exponential Behavior

SYNOPSIS:

Proper exponential behavior of the wave functions used in STM modeling can be sometimes be obtained by scaling-out the Gaussian basis function exponents of standard valence basis sets. Other times it is necessary to optimize the exponent of the diffuse function using excited states.

The basis sets normally used in Hartree-Fock, GVB, and higher correlation methods are optimized for predicting the bonding and electronic excitation chemistry of molecules. STM calculations have somewhat different requirements from these types of calculations because we are often interested in the tails of wave functions. Therefore, care must be taken to use a basis set that is augmented to display proper exponential decay. Here we demonstrate that our basis sets result in wave functions that have the proper exponential behavior.

We can compare Figure 4 to Figure 25 to see that the unmodified basis set performs poorly for describing the vacuum decay of the Ni states. As a further test, we present fits of the functions in Figures 4 and 25 over a different distance range, from 6 bohr to 11 bohr. Notice that for the augmented basis set, the fit is good and mutually consistent in both distance ranges. This is not true for the unscaled basis set.

We found that for the Ni case, a simple scaling was adequate. For the Xe case we tried scaling using two additional *s*-functions at 0.418 and 0.117 along with a *p*-function at 0.0416, however, the exponential behavior was poorly described beyond the Xe. This could be due to the imbalance we left in the *s* and *p*-functions since we did not incorporate an additional *p*-function at 0.0138 corresponding to the *s*-function at 0.0117. Regardless, we found that good results were obtained with just a diffuse *s*-function that was optimized

for excited Xe (triplet Xe at the Hartree-Fock level). In fact, for the cases with Xe over the Ni surface, we find that the Ni diffuse functions are not needed. Figure 28 shows the Ni HOMO decay through a Xe atom at 3.4 Å above the origin just as in Figure 5 but with no Ni diffuse *s*-function. The fit parameters for the reduced basis set are good and nearly identical with those of the augmented Ni basis.

In conclusion, it is necessary to check the exponential behavior of the wave function before relying on a particular set of basis functions; however, it is not difficult to get good exponential decay behavior with Gaussian basis functions.

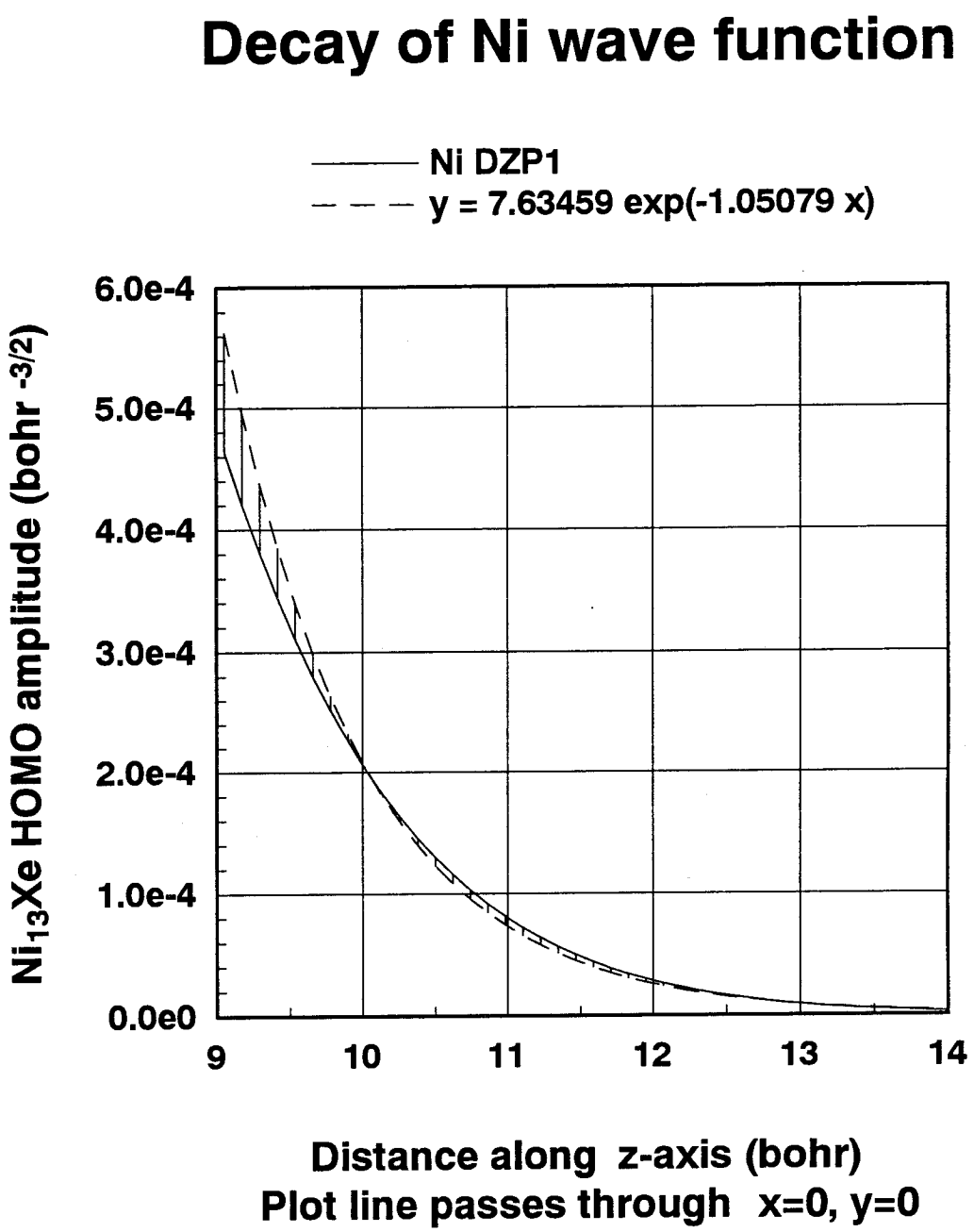


Figure 25:

Decay of Ni HOMO from 9 to 14 bohr using the standard Ni DZP1 basis set. Shown with an exponential fit which indicates poor exponential decay behavior.

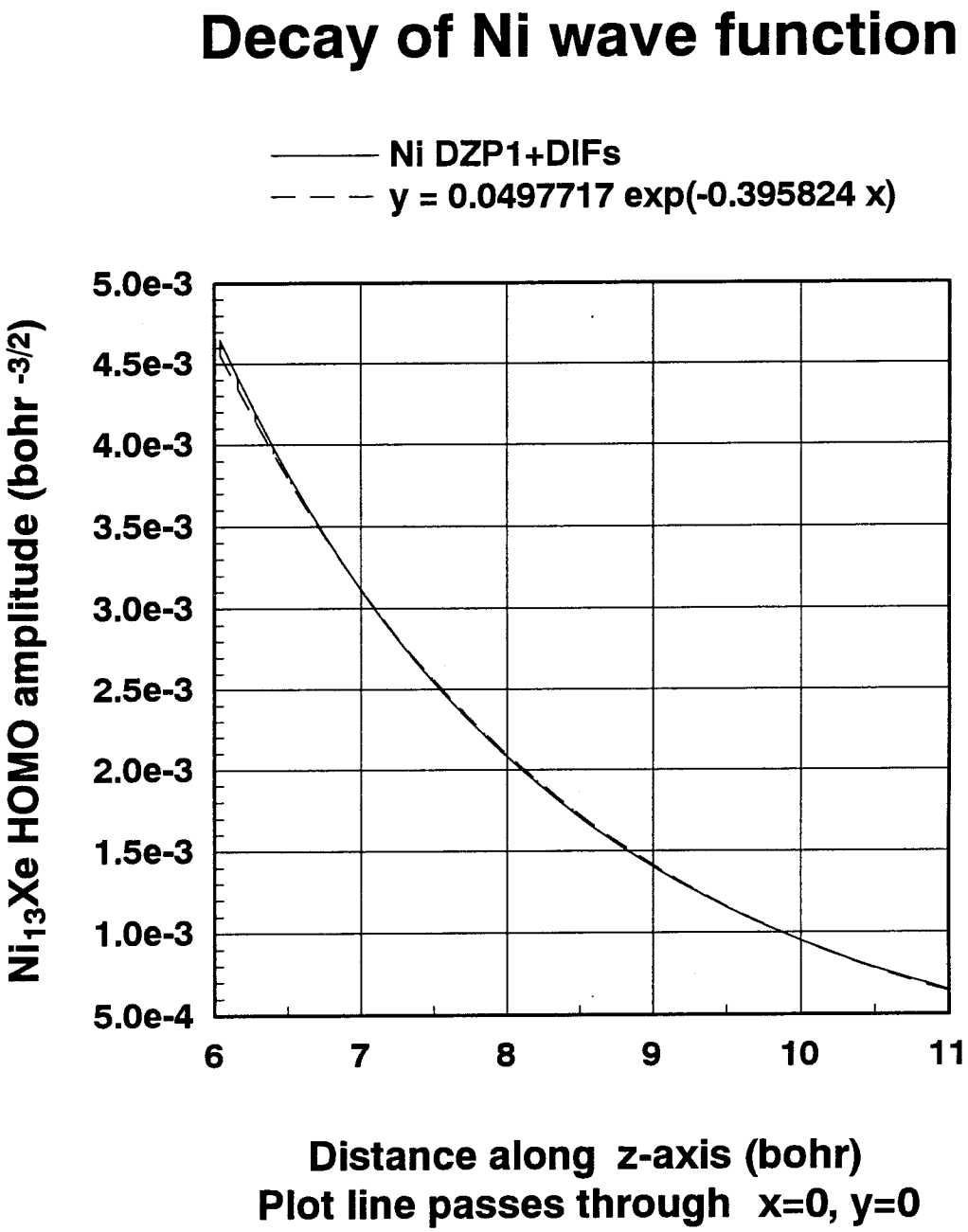


Figure 26:

Analogous to Figure 4 but over the range 6 bohr to 11 bohr.

The match in the fit parameters indicates proper behavior over the entire region.

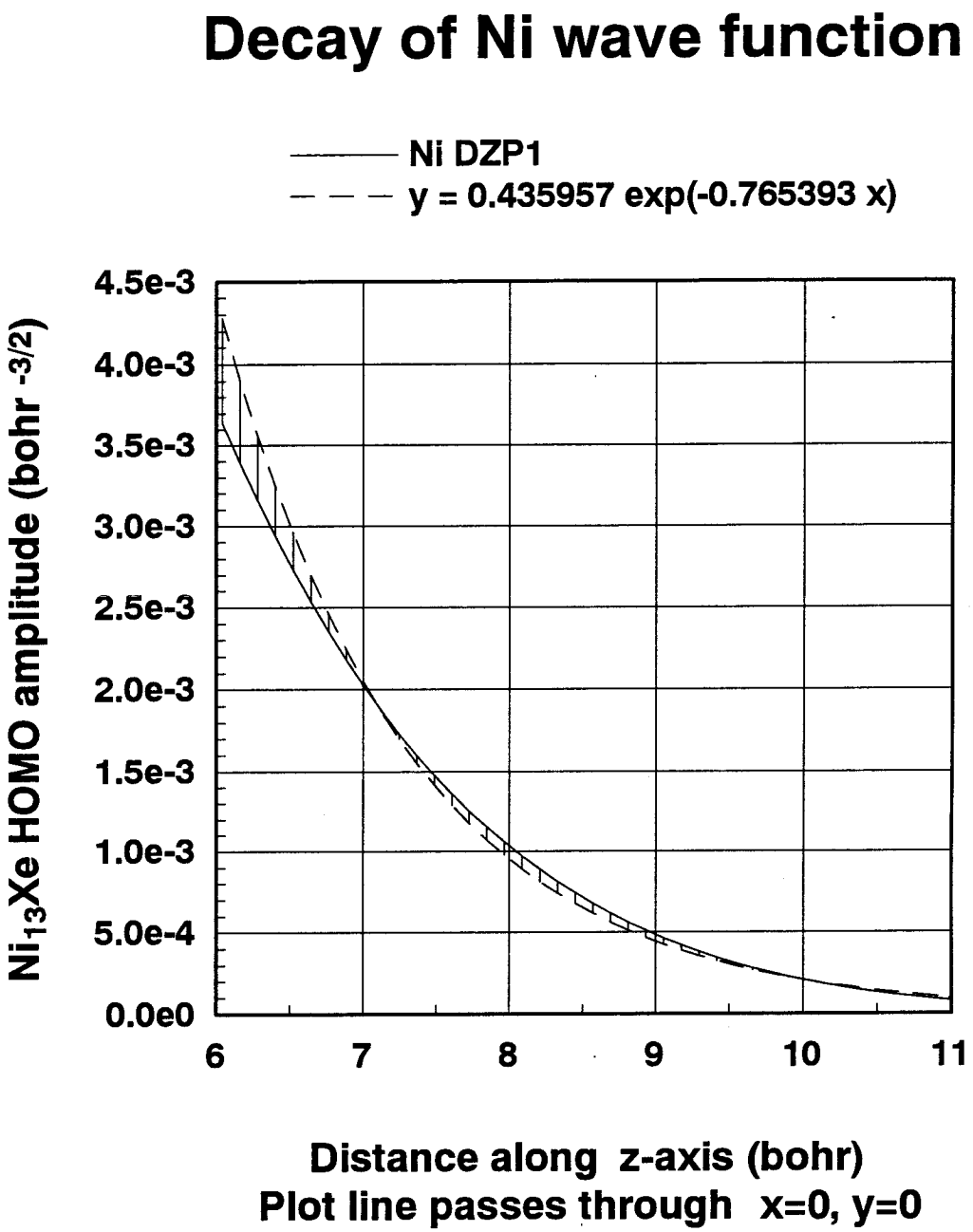


Figure 27:

Analogous to Figure 25 but over the range 6 bohr to 11 bohr.
The fit is poor and the parameters are considerably different to those of Figure 25. This indicates a poor description of the exponential behavior of the Ni wave function.

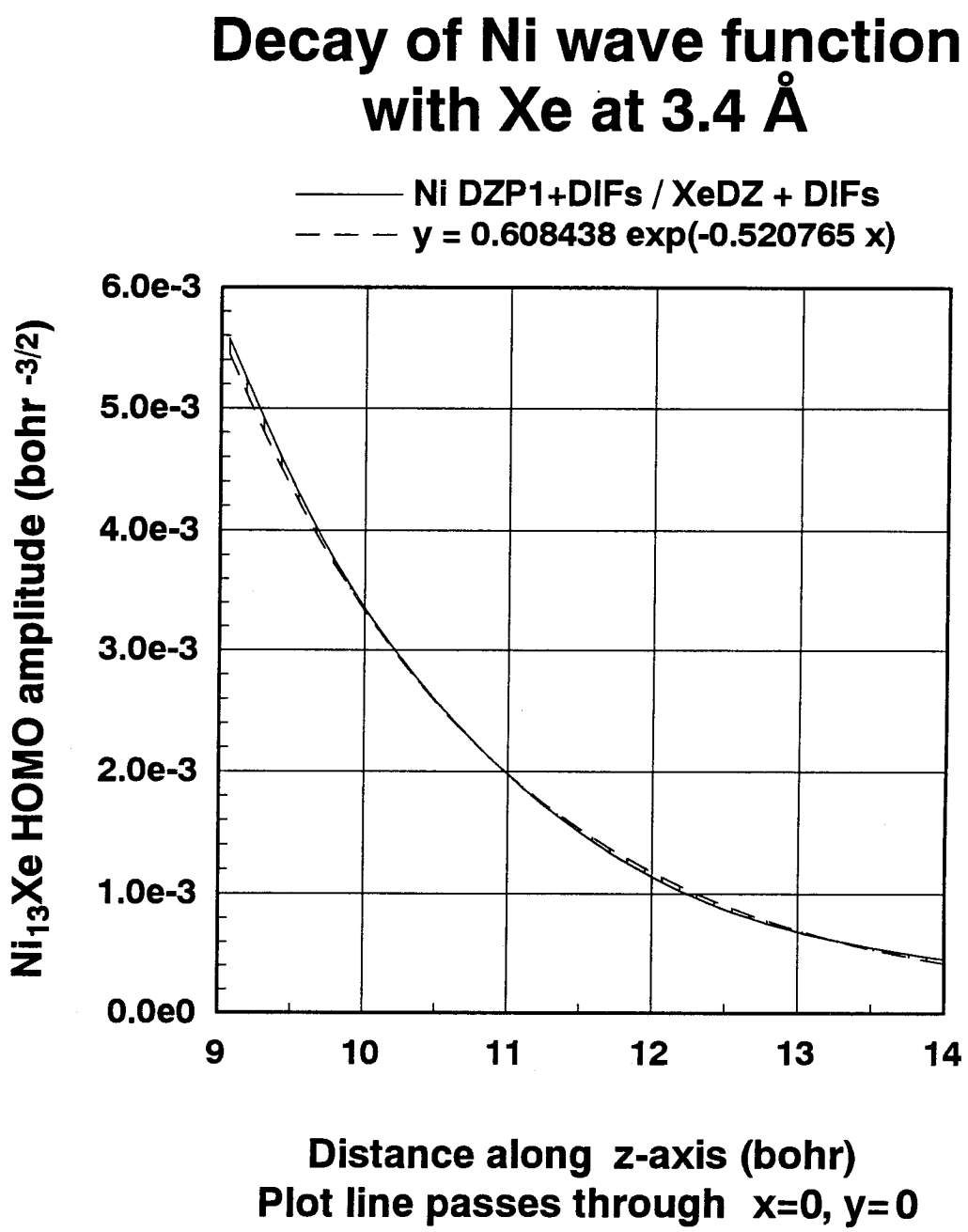


Figure 28: Decay of the Ni HOMO through a single Xe atom at 3.4 Å (6.425 bohr) above the central cluster Ni. This is analogous to Figure 5, however, the Ni basis set is lacking the extra scaled out *s*-function. The results show that the extra Ni function is not needed when the Xe is present and properly described.

3.4.4.2. Eliminating Basis Set Superposition Error

SYNOPSIS:

By performing a Ni cluster calculation with Xe basis functions, but without the Xe atom, we eliminate the possibility that the Ni HOMO perturbation resulted from basis set superposition error.

Here we check the possibility that the Ni wave function is using basis functions from the Xe regardless of the presence or absence of the Xe nucleus. If it were occurring, the use of these basis functions would have physical significance: the extra degrees of freedom are significantly energetically favorable. However, this would invalidate any conclusions based on the presence of adsorbates perturbing the Ni wave function. Figure 29 shows three lines. In each case the Ni basis set was Ni DZP1. The Xe atom at infinity (effectively a bare Ni cluster) serves as the reference. Next, we completed two new calculations using the Xe DZ + DIFs basis set. In one case, the Xe nucleus and electrons were present; in the other case these were absent but the basis functions for the Xe electrons were present.

In the case of the missing Xe atom, the Xe basis functions contribute slightly to the decay properties of the Ni HOMO wave function. The result is a better description of the Ni decay in to the vacuum. However, when the Xe is actually present, the perturbation is large and qualitatively different. Therefore we conclude that the effects we are studying are real and due to the presence of the adsorbate and not the extra degrees of freedom afforded by the basis functions associated with the adsorbate.

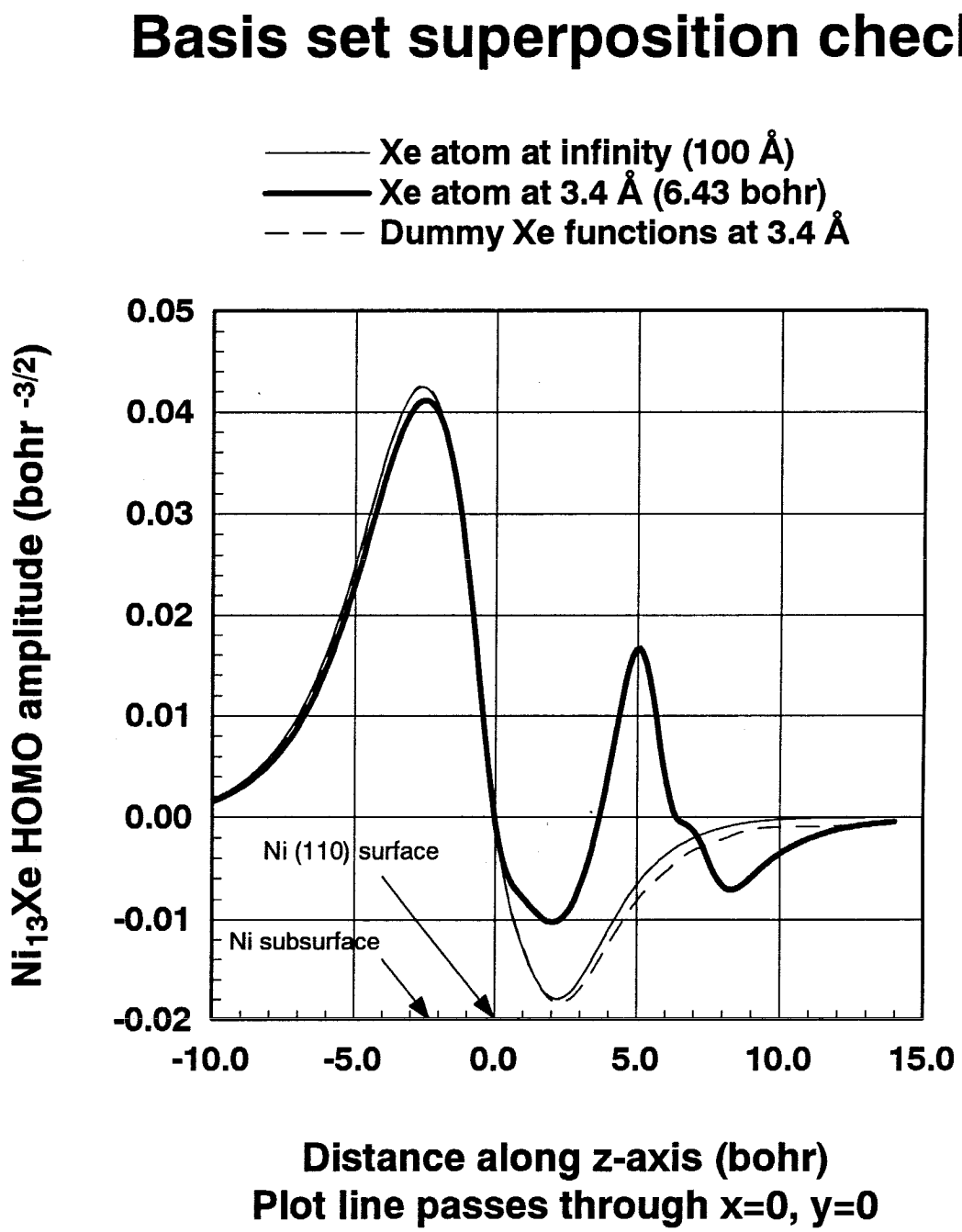


Figure 29:

Basis set superposition check. In each case the Ni basis functions were Ni DZP1 and the Xe functions were Xe DZ + DIFs.

4. STM of Large Adsorbates on Graphite

In recent years, the STM has come into its own as a sensitive technique for the imaging of individual molecules on surfaces. Perhaps some of the most striking work in adsorbate imaging has been the visualization of DNA strands on graphite and other substrates as reported by several groups.[30,31,32,33,34,35] Critical to making further progress in this area will be the ability to predict what features will distinguish base pairs in the STM image. In particular, the ability to intelligently design a base labeling scheme may become the limiting step in successful DNA sequencing by STM.

4.1. Introduction

In this section we study the case of *n*-butyl benzene physisorbed to a graphite mimic. Our goal is to gain insight into the following questions:

- What is a plausible imaging mechanism for this system?
- Can we differentiate chemical units on the adsorbate?
- What are good computational models for these types of systems?

---

[30]M. G. Youngquist, R. J. Driscoll, T. R. Coley, W. A. Goddard, III, J. D. Baldeschwieler *J. Vac. Sci. B* **9** 1304-1308 (1991). Work done on graphite.

[31]N. J. Tao, J. A. DeRose, S. M. Lindsay *J. Phys. Chem.* **97** 910-919 (1993). Work done on Au(111).

[32]A. Cricenti, S. Selci, A. C. Felici, R. Generosi, E. Gori, W. Djaczenko, G. Chiarotti, *Science* **245** 1226 (1989).

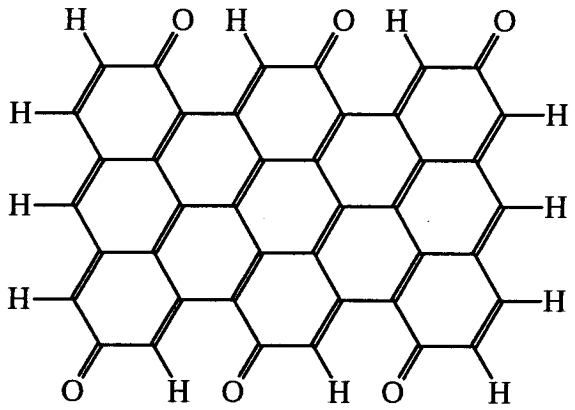
[33]G. Lee, P. G. Arscotti, V. A. Bloomfield, D. F. Evans, *Science* **244** 475 (1989).

[34]D. Keller, C. Bustamante, R. W. Keller, *Proc. Natl. Acad. Sci.* **86** 5356-5360 (1989).

[35]C. R. Clemmer, T. P. Beebe, S. M. Lindsay, K. W. Hipps, J. D. Baldeschwieler *Scan. Micros.* **6** 319-333 (1992). Review.

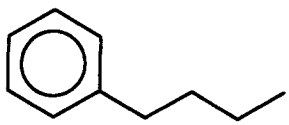
4.2. Model for Graphite/Adsorbate System

The following $C_{42}O_6H_{12}$ molecule was constructed as a model for graphite:



The hydrogens and oxygens are used to properly tie off the dangling bonds of the graphite fragment. As a result, we are also favoring one of the resonance states of the graphite. The consequences in the predicted image will be discussed.

An adsorbate molecule of *n*-butyl benzene was chosen because of the presence of both an aromatic and aliphatic group. The *n*-butyl benzene molecule, show below, was docked to the substrate mimic using the BioGraf program.[36]



[36]BioGraf is a product of Molecular Simulations, Inc., Waltham MA. The force field used was Dreiding II.

An energy minimum that positioned the adsorbate near the center of the substrate was chosen (see Figure 30). The coordinates are given in Table 8. These coordinates were used as input for the pseudospectral Hartree-Fock program.[60] The total system has 386 electrons occupying 193 Hartree-Fock molecular orbitals.

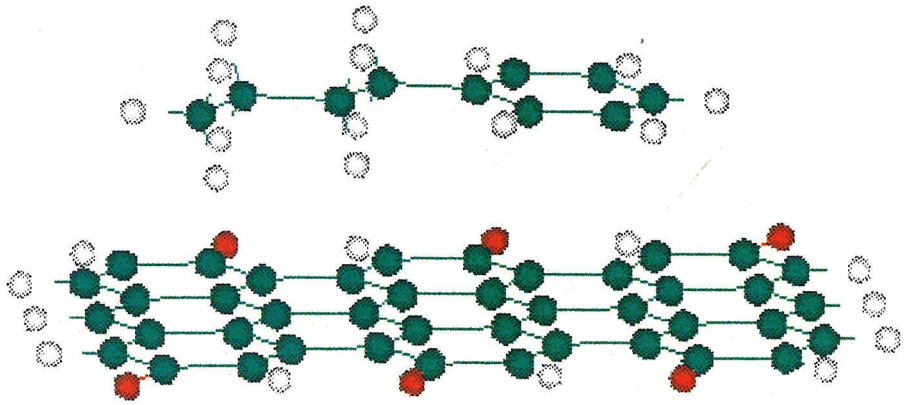


Figure 30:
Side view of the *n*-butyl benzene on graphite model:
carbon in dark green, oxygen in red, hydrogen in white.

4.3. Results

We will be examining the process of removing electrons from the surface/adsorbate system. In the STM experiment a bias voltage selects which states of the sample will contribute to the tunneling current. Although both tip-positive and tip-negative bias voltages have been used, in many cases of organics on graphite images are acquired using V_{tip} in the +1 to +2 volt range, and sometimes higher.[37]

Three energy ranges of occupied states are used to compute electron densities available for tunneling and predict images. The first state available for tunneling out of the sample in our model is, of course, the highest occupied molecular orbital (HOMO). The first range of contributing states we explore is a window of the highest seven states. These orbitals all have energies within 2 eV of the HOMO. The second range consists of the highest eleven states and comprises an energy window of 4 eV. Finally, we compare these images to the total density of states and discuss the relationship of the STM images to the underlying atomic topology and the total electronic density.

4.3.1. Fast-Scan Images

SYNOPSIS:

Plots of the densities of states available for tunneling are shown in a plane above the adsorbate and another plane below the substrate of the model. These images correspond to what would be expected from fast-scan (constant height) STM experiments of the adsorbate and bare surface, respectively. The results successfully predict images of graphite and confirm the ability of the STM to selectively image chemically distinct regions of the adsorbate molecule.

---

[37] J. J. Breen, G. W. Flynn, *J. Phys. Chem.* **92** 6825 (1992).

Figure 31 shows the total electron density (bohr^{-3}) in a plane z = -3.5 Å, or roughly 5.2 Å above the graphite and 1.5 Å above the *n*-butyl benzene ring on the adsorbate side of our model. This image shows that the surviving density of tunneling electrons at this height exists primarily above the aromatic ring with some delocalization to C_1 of the alkane chain.

There are several interesting features of this image. First, we note that the ring actually appears as a doughnut shape. This is what we expect from the chemistry of aromatic rings, but it is not what has been observed experimentally. The reason is simply that our image represents what would result from an infinitely sharp point-tip sampling electron density at each point in the scan plane. In the experiment, the image that results is better described by a convolution of the wave function we are plotting with the wave function of the tip.

To show the affect of a finite tip, we plot the same density of states in Figure 32 with a 25-point exponential filter applied.[38] The algorithm for generating the filtered data at each point is

$$d_{pq}^{filtered} = \sum_{i=p-2}^{p+2} \sum_{j=q-2}^{q+2} d_{ij} \exp\left(-1.0\left|l_{ij} - l_{pq}\right|\right) \qquad (4.1)$$

where $|l_{ij}-l_{pq}|$ is the distance in bohr between the location of grid points ij and pq, and d_{ij} is the density at the grid point ij. This is the simplest filter that will generate an exponentially decaying contribution from each neighboring site. The decay constant can be varied from unity to represent a more diffuse or a sharper tip acceptor, but there is not much more insight into instrument resolution to be gained from this particular case.

---

[38]See the software section of this thesis for a description of the general tools we have developed and how they can be used to test various filter functions.

In a model system with adjacent adsorbates, one could calculate additional convolutions to estimate the effective tip radius needed to allow the STM to resolve the adsorbates. We have not explored such cases here but have developed the techniques and tools which can be applied to these questions.

Our plots of non-convolved sample density correspond to the hypothetical infinite resolution case. The contours of electronic states will generally be spread over atomic centers, and the diffusiveness of states increases with the height of the imaging plane chosen. If a feature cannot be resolved in this infinite resolution type of plot, it will not be resolved experimentally.[39]

Another interesting feature of this particular system is the increase in density over the aromatic carbon attached to the alkane chain, and the two carbons *ortho* to the point of attachment. We will see later that this effect persists to the deeper energy window we sampled. If we take a symmetric distribution of π-electrons about the ring as a reference and recall that our image is sampling the higher energy π-electrons, we conclude that the energy of states located above these three carbons is raised with respect to the rest of the ring. Two possible reasons for energy shift are 1) the two *ortho* carbons are located almost directly on top of substrate carbons, and 2) there is an anti-bonding interaction between the π-ring states and the π-combination of the two C-H bonds of the first alkane carbon. Figure 33 shows an example orbital with this character. This phase of interaction exists in several other of the highest energy states and contributes to the enhanced density at high energy near the alkane side chain. Note that a typical tungsten tip with a radius of roughly 1.5 Å is not likely to resolve this feature.

---

[39]This is based on the assumption that the tunneling current is proportional to the local density at the Fermi level. In the section on Bardeen matrix elements we discuss a situation where the resolution may be enhanced relative to the Tersoff-Hamann predicted image.

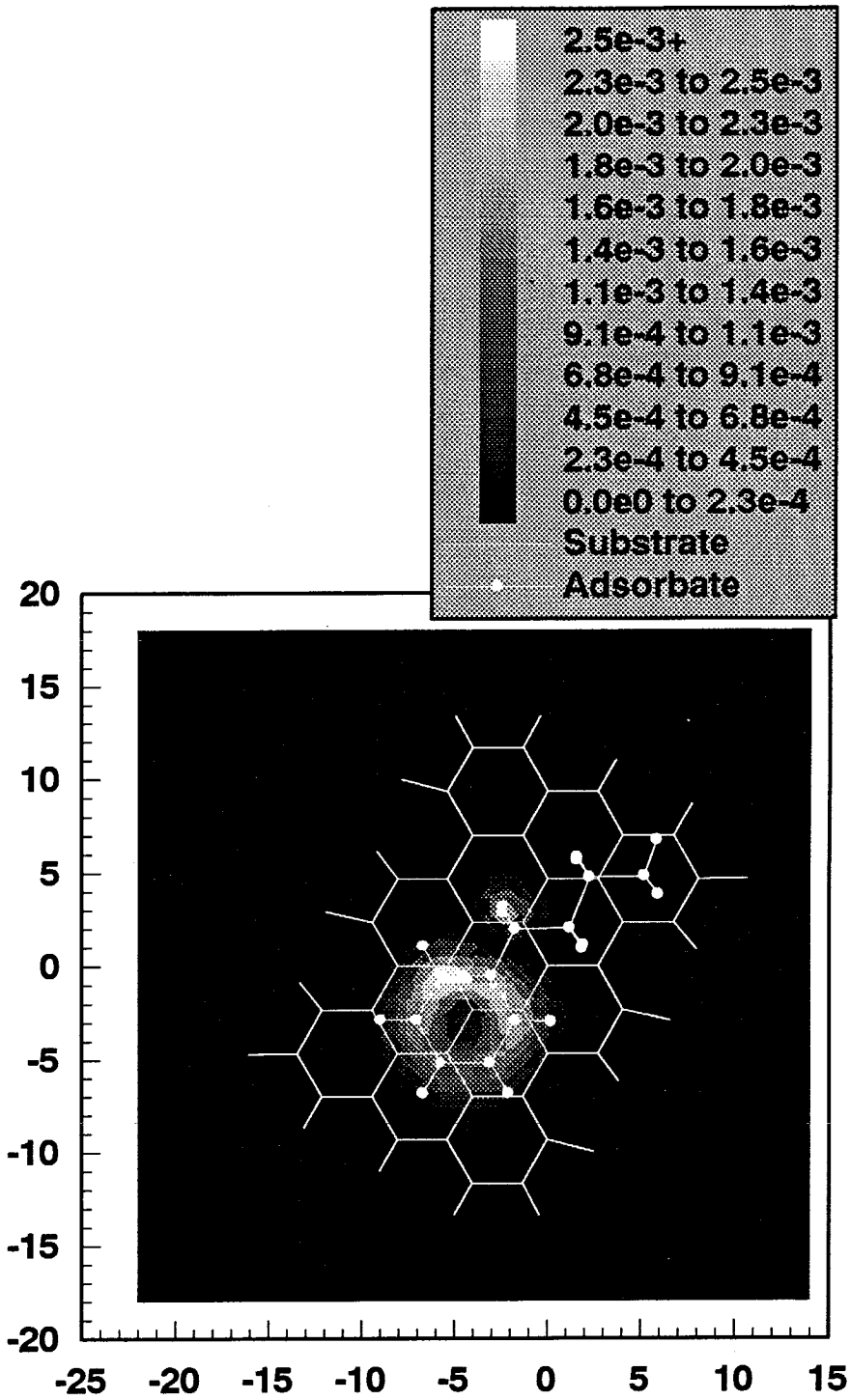


Figure 31: Density of states for orbital energies -0.34693 to -0.27281 hartree (a 2 eV window) shown with the molecular frame. Plot plane is z = -3.500 Å or 5.168 Å above the graphite and roughly 1.5 Å above the plane of the adsorbate molecule. Contours are in units of bohr^{-3}.

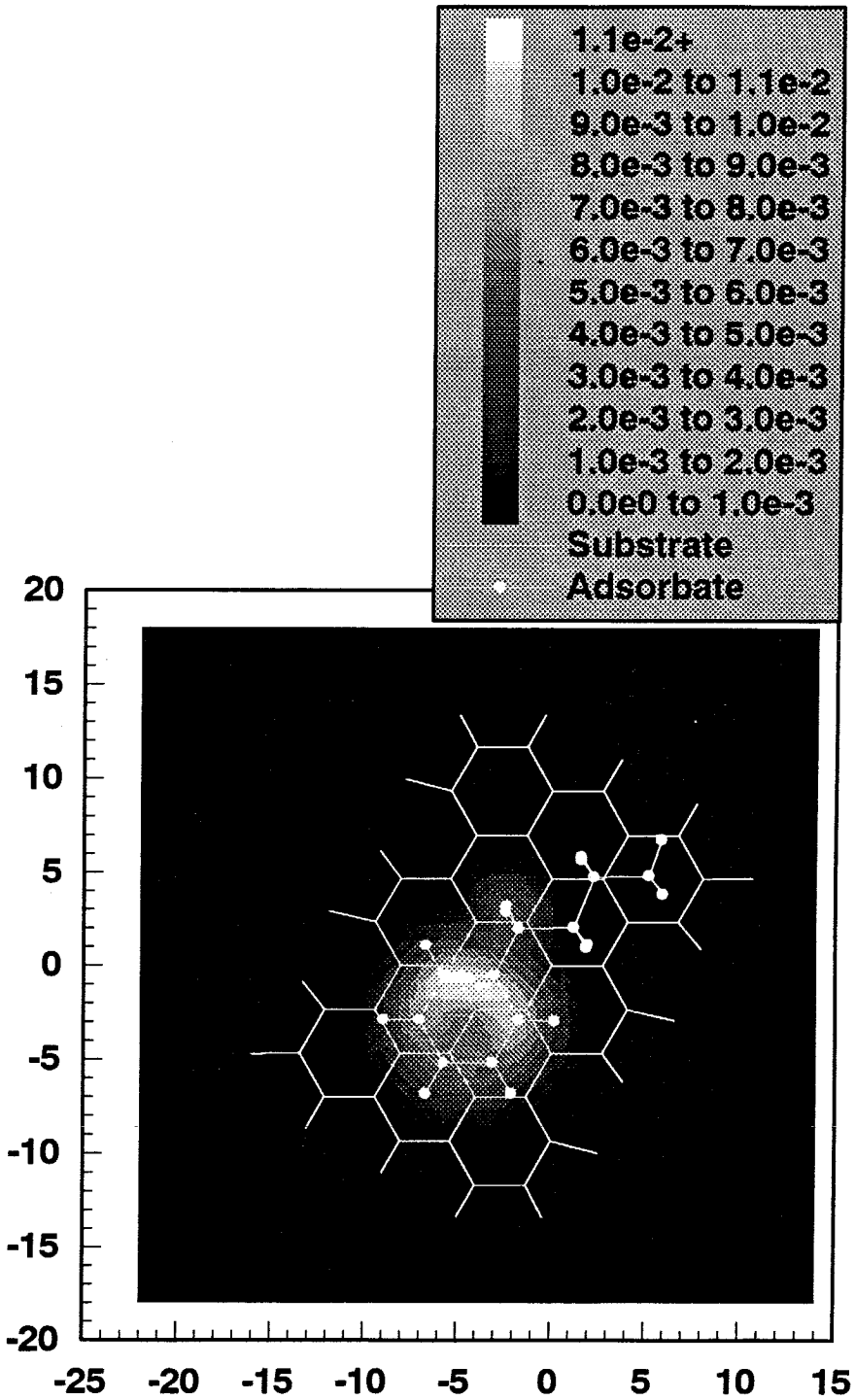


Figure 32:

Same as Figure 31 but with a 25-point filter applied. See text.

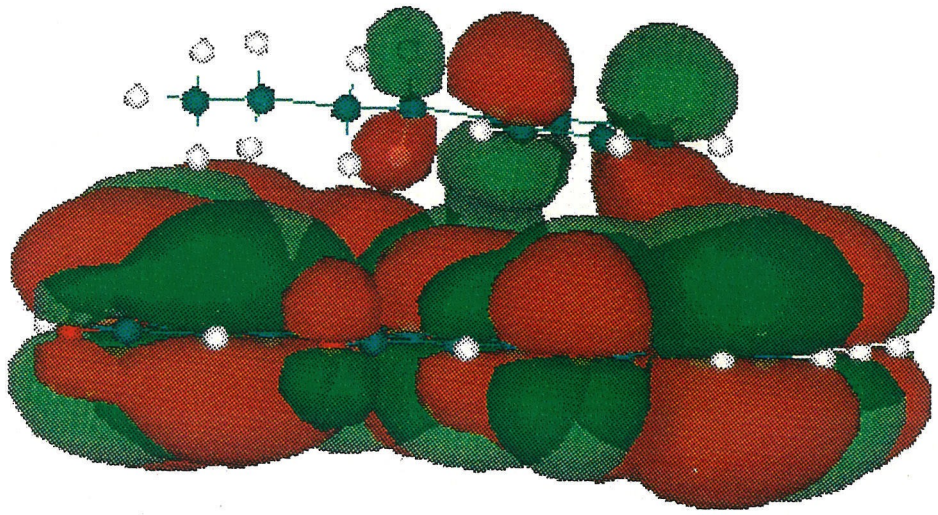


Figure 33: The state third below the HOMO showing the anti-bonding interaction between the aromatic ring and the C-H bonds of the first alkane carbon. These high energy states are responsible for the increased density near the side chain carbon. Isosurface is +/- 0.003 $bohr^{-3/2}$.

In Figure 34 we have plotted the same set of states previously discussed. However, we have taken a plane *below* the graphite (i.e., on the side opposite the adsorbed molecule). The idea is that this plane should give us a prediction of the bare graphite surface without needing to do another large computation. Figure 35 shows the same set of states with the 25-point filter applied. There are several interesting conclusions we can draw from these plots.

First, we should point out that images of graphite are notorious in the STM community for generating a variety of image types depending on the conditions. Many patterns can be ascribed to double tips.[40] The images we have generated from our *ab initio* calculations obviously correspond to single tip imaging, yet we do not see perfect 6-fold symmetry. The overlaid carbon framework (and subsequent three-dimensional renderings) reveals that the nature of the states being imaged is biased to the π-space of the graphite resonance structure we have favored with the terminating oxygens. Such biases can occur in the experiments when impurities or defects (edges) break the hexagonal symmetry.

We also observe that the C=O bonds are not imaged in the energy window selected. This is because the more electronegative oxygen has lowered the energy of nearby states below our window of states. This affect of oxygen has also been observed experimentally with O atoms chemisorbed to Ni (100).[12] In this system the O atoms appear as depressions in the constant current mode. Our model predicts a similar effect for an oxygen containing molecule.

Another effect of our finite cluster is seen in the bright spots over the two double bonds at the top and bottom of the cluster. These two are the only C=C double bonds

---

[40]See H. A. Mizes, J. S. Foster, *Science* **244** 559 (1989) for a particularly interesting discussion of one effect on graphite. Also, references therein.

with a terminal hydrogen on each carbon. As a result, these bonds are less stable than their "bulk" counterparts (in the center of the cluster). Although, again, from the standpoint of modeling graphite, this is an artifact of our system. Nevertheless, these states at the edge of the cluster are a good electronic description of a double bond type *different* from the other double bonds in the system. The ability to distinguish it from the neighboring bonds gives a sounder basis to the hypothesis that unsaturated carbon chains may be distinguished from aromatic groups under the right conditions.

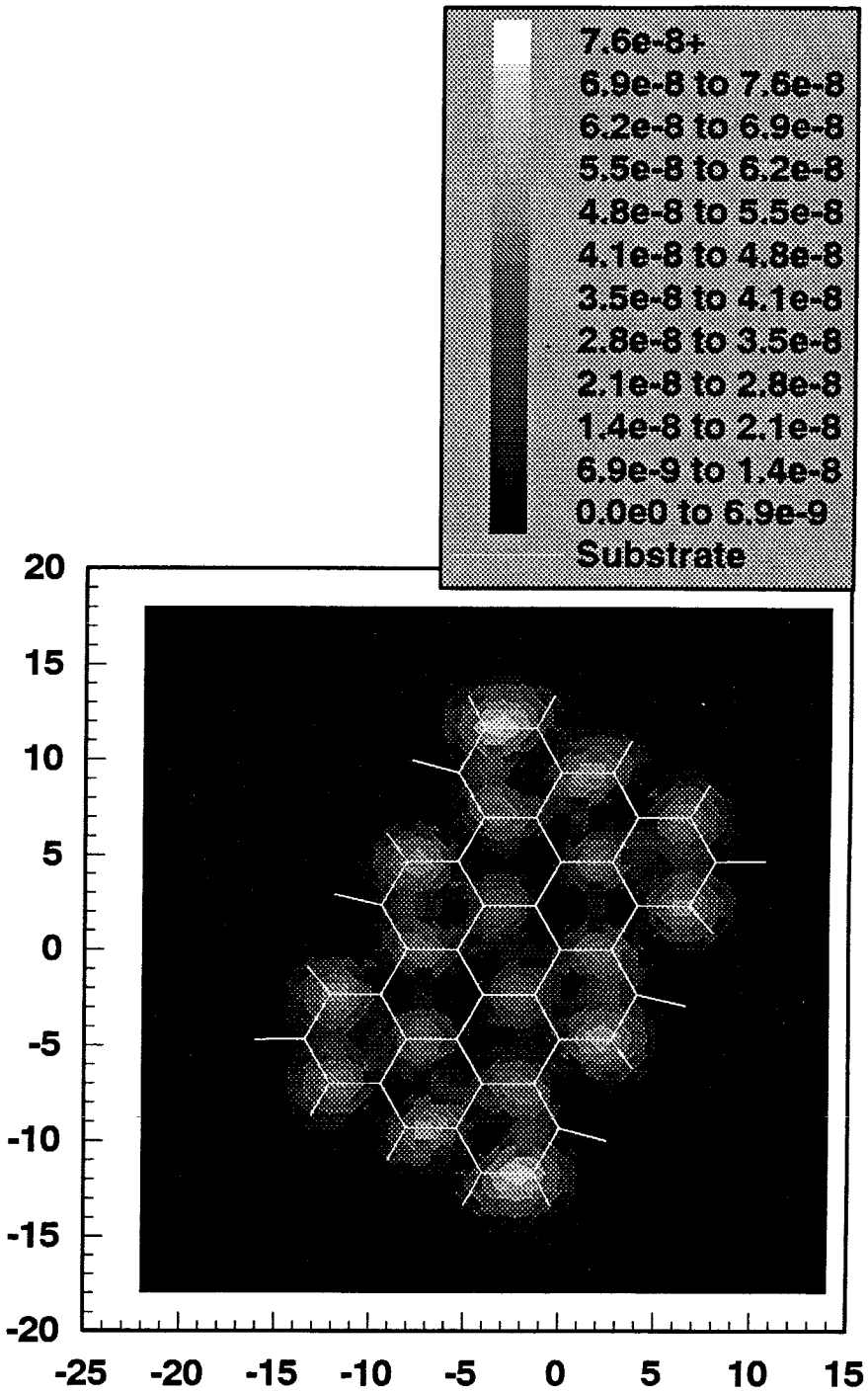


Figure 34:

Density of states for orbital energies -0.34693 to -0.27281 hartree (a 2 eV window) shown with the substrate framework. Plot plane is z = 5.168 Å or 3.5 Å from the graphite plane on the side *opposite* the adsorbate molecule. Contours are in units of bohr^{-3}.

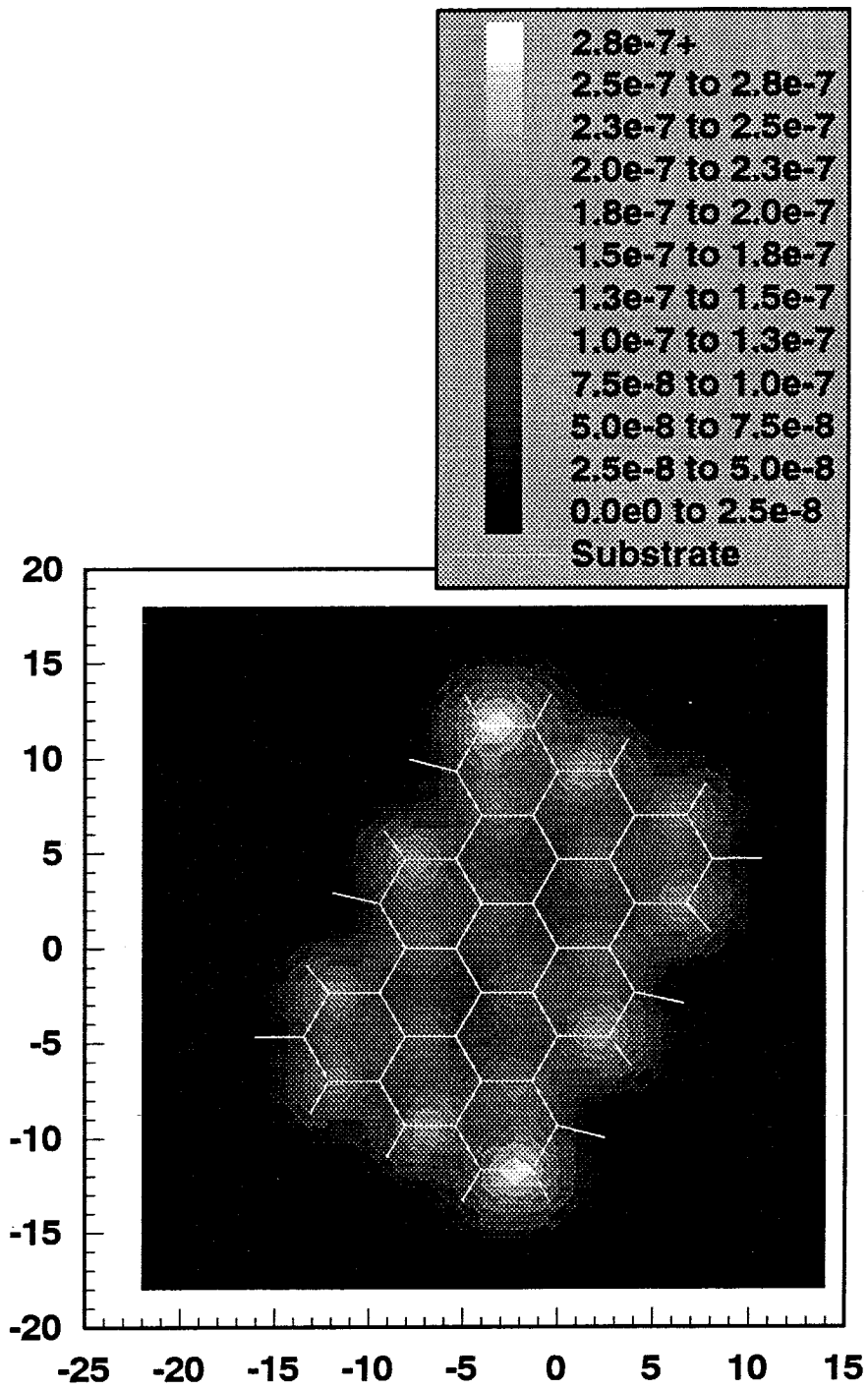


Figure 35:

Same as Figure 34 but with a 25-point filter applied. See text.

To check the sensitivity of our predicted images to the bias voltage we doubled the energy window of states selected to contribute density in the fast-scan images. Figures 36 and 37 show the unfiltered and filtered images corresponding to those of Figures 31 and 32. This time the orbital energies included eleven states from -0.41941 to -0.27281 hartree (a 4 eV window). The plotted densities show no qualitative change in the predicted images. Quantitatively, the maximum density on the grid goes from 0.00312 electrons/bohr^3 to 0.00315 electrons/bohr^3 indicating that the new states are adding less than 1% to the total density above the adsorbate.

Figures 38 and 39 show the images with the expanded energy window. Compare these with Figures 34 and 35. A qualitative change in the image is noticeable as a filling out of the double bond space and the beginnings of imaging along the single bond axes. Quantitatively, the maximum density has risen from 8.31×10^{-8} electrons/bohr^3 to 1.14×10^{-7} electrons/bohr^3, a rise of 38% relative to the original maximum.

From these images we conclude that while some change in the fast-scan image above the graphite is seen, the adsorbate image is stable over a large range of bias voltages. The reason for this is clear from the orbitals of the system: there is essentially no additional adsorbate state density in the broader energy window. The next available adsorbate states are the C-H bonds and we need to go to much deeper energies to access these.

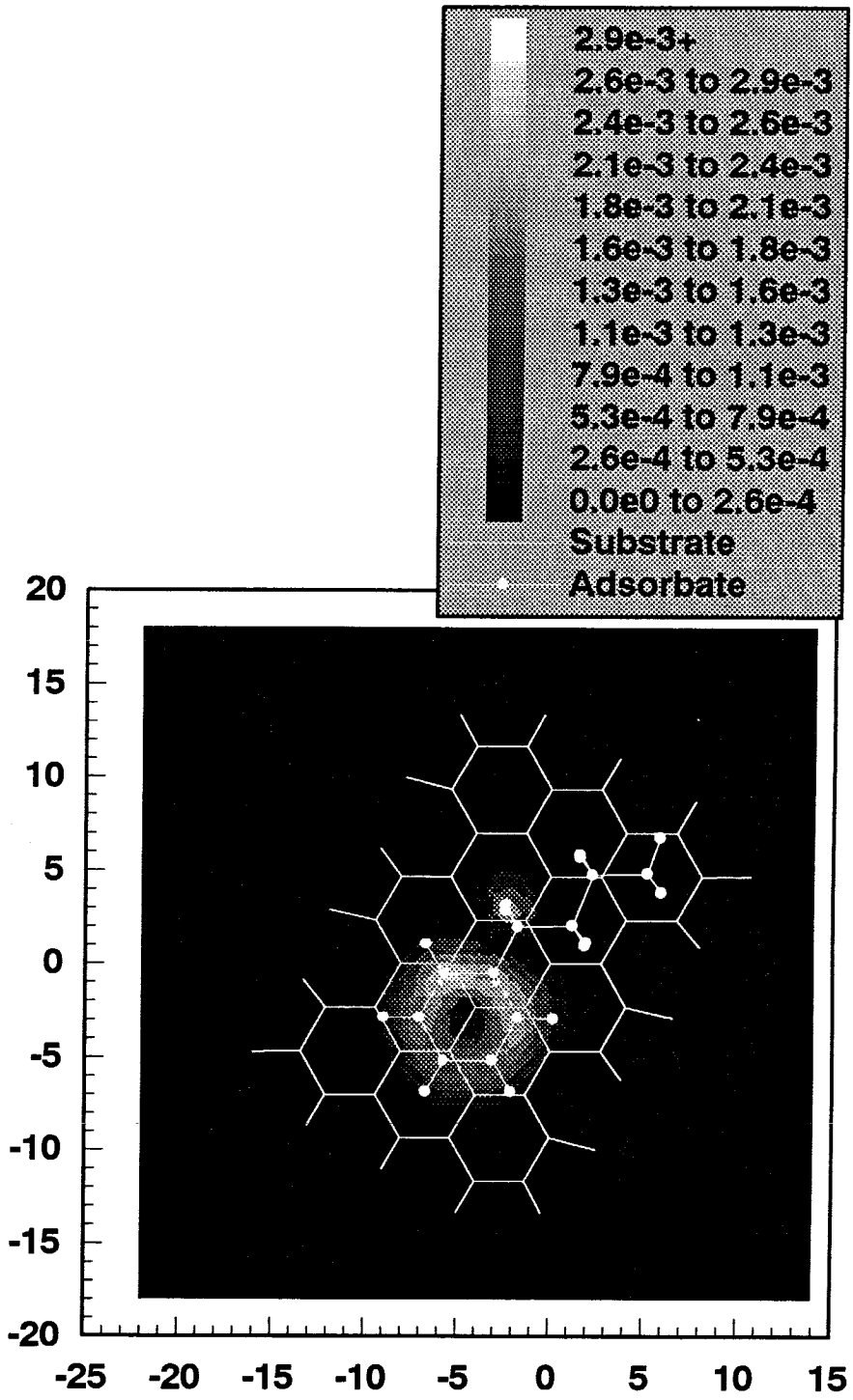


Figure 36: Density of states for orbital energies -0.41941 to -0.27281 hartree (a 4 eV window) shown with the molecular frame. Plot plane is z = -3.500 Å or 5.168 Å above the graphite and roughly 1.5 Å above the plane of the adsorbate molecule. Contours are in units of bohr^{-3}.

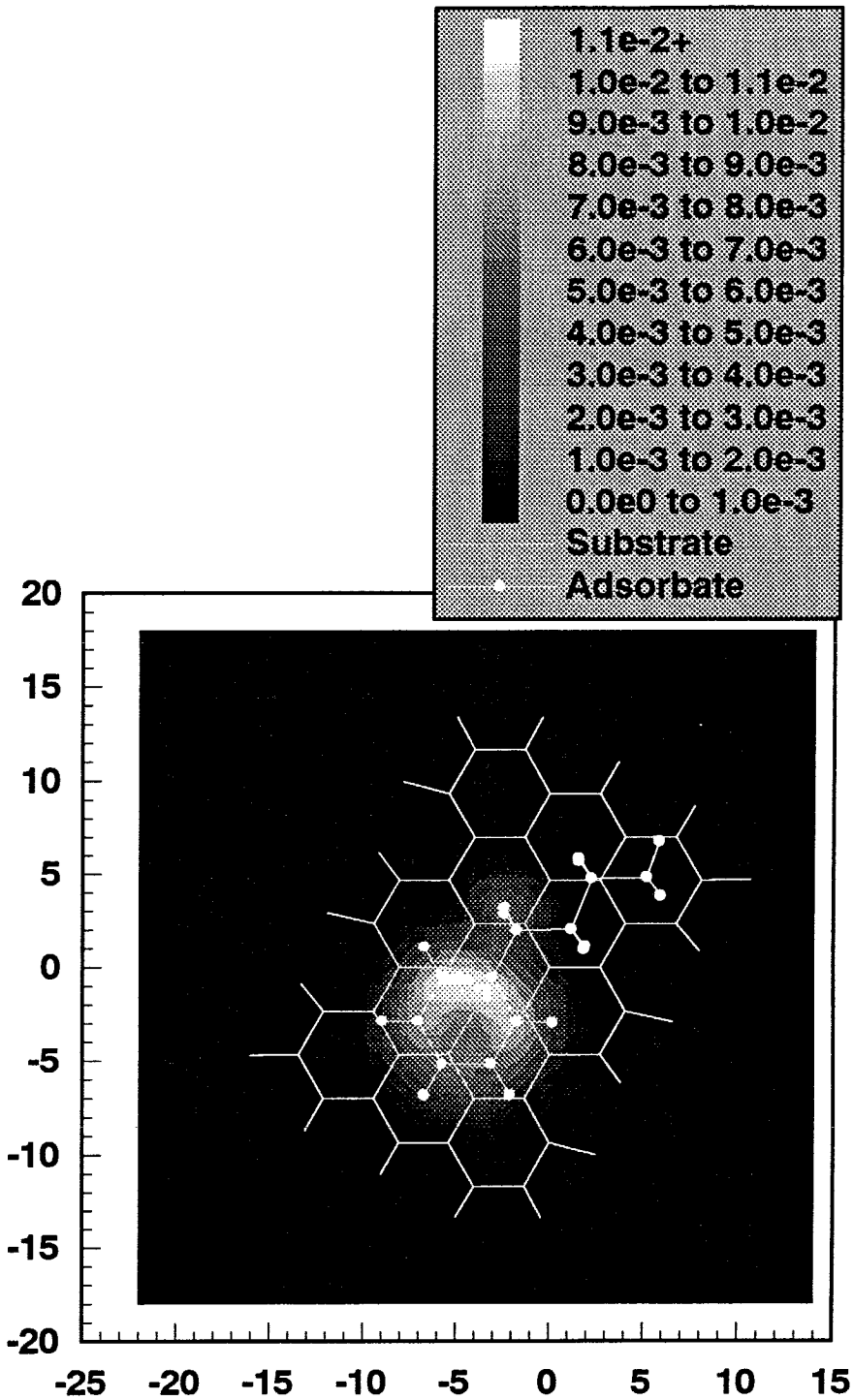


Figure 37:

Same as Figure 36 but with a 25-point filter applied. See text.

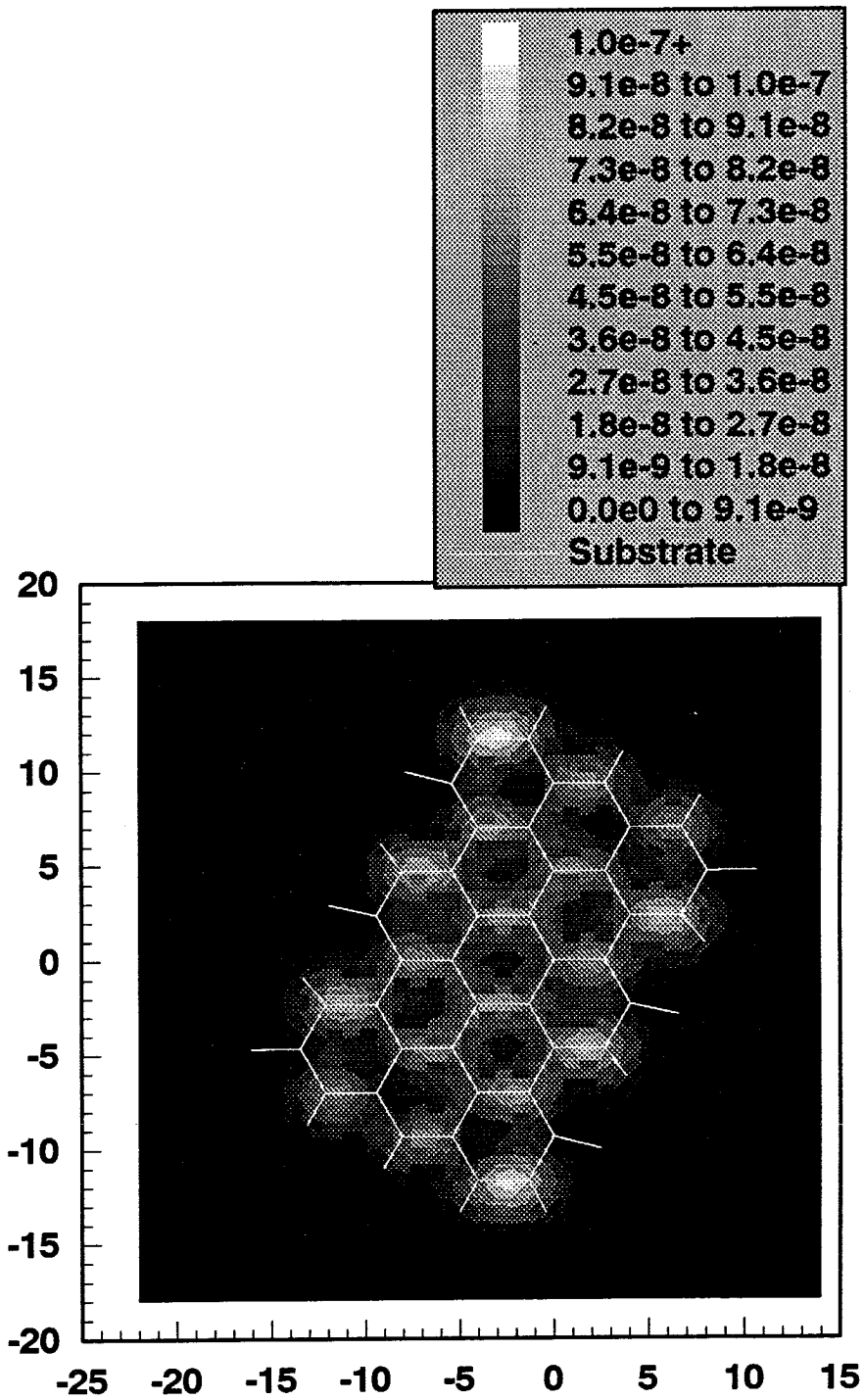


Figure 38: Density of states for orbital energies -0.41941 to -0.27281 hartree (a 4 eV window) shown with the substrate framework. Plot plane is z = 5.168 Å or 3.5 Å from the graphite plane on the side *opposite* the adsorbate molecule.
Contours are in units of bohr^{-3}.

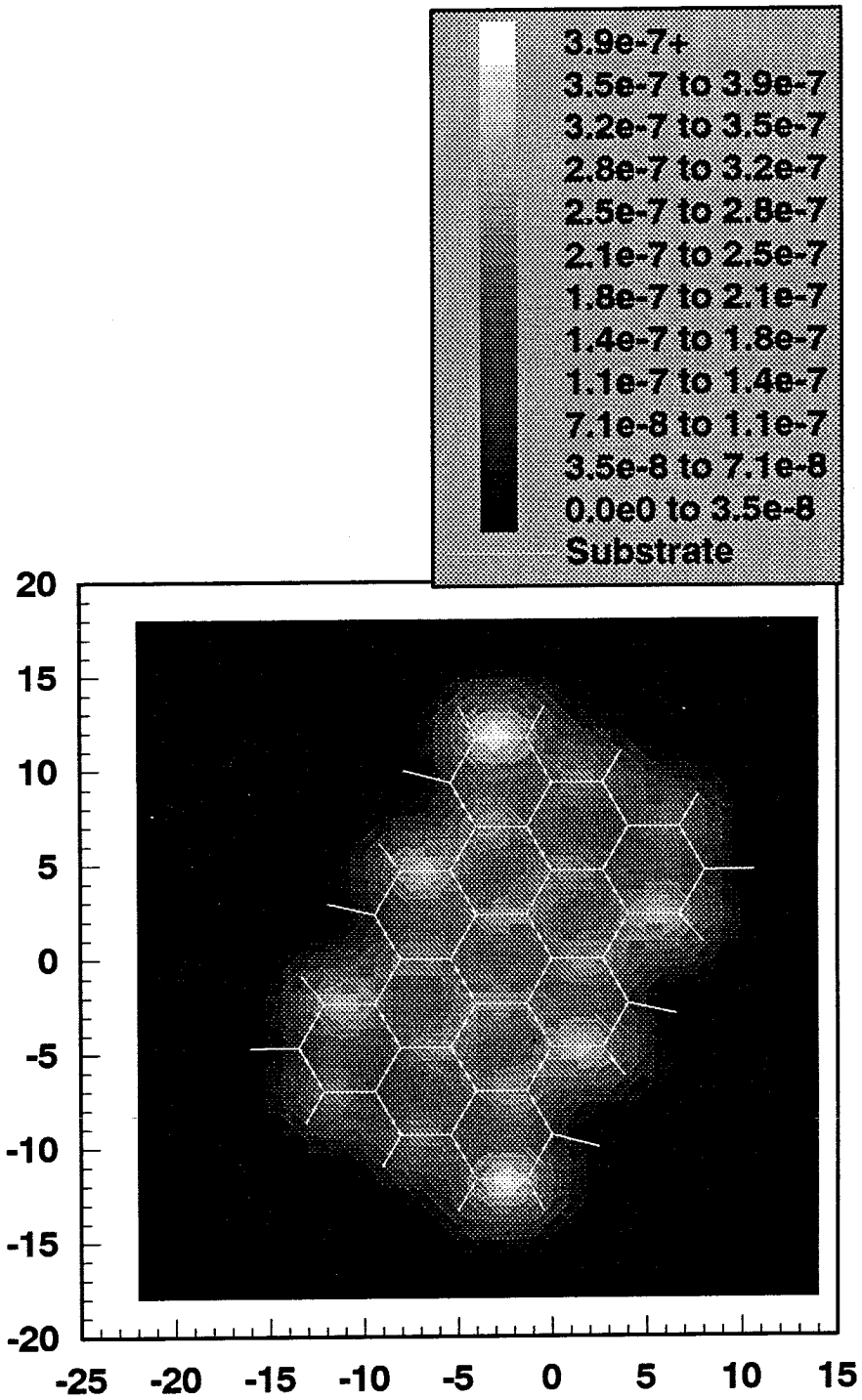


Figure 39:

Same as Figure 38 but with a 25-point filter applied. See text.

In many areas of chemistry we are accustomed to visualizing molecules in terms of the locations of the nuclei or in some cases in terms of Van der Waals radii. Space filling models are generally related to the total electronic density about the molecule. As has been pointed out, however, the STM produces images that are a reflection of the electronic density belonging to a select energy window. Therefore, it is instructive to compare the previous images with those that would arise if density from *all* states were imaged. Therefore, in the same planes as before, we plot the image that would be generated by a hypothetical fast-scan STM experiment that sampled all valence states. (The core states, O and C 1*s* orbitals, contribute no significant density at the distance of our plot plane.)

Figures 40 and 41 show the unfiltered and filtered densities just above the carbon plane of the adsorbate molecule while Figures 42 and 43 show the density 3.5 Å above the other side of the graphite model. The graphite image shows overall similarity to the previous images, though it displays a flatter current modulation when the filter is applied. The flattening results from the lesser relative concentration of density in the C=C bond space.

In contrast, the adsorbate image is completely changed. The reason is that we have chosen a plot plane very close to the plane of the adsorbate molecule's highest H atoms. This choice of plot planes demonstrates that even when we attempt to maximize the possibility of imaging the H atoms by choosing a plane that should maximize the density due to electrons near the hydrogens, the restricted energy window imaged by the STM renders the H atoms *completely* invisible.

Recognition of this fact is important because it has direct bearing on whether we could image such a system in the actual experiment. Consider a system where instead of an isolated molecule we have an ordered overlayer of *n*-butyl benzene molecules. Also,

assume for the moment that the aromatic rings remain parallel to the substrate surface. If we start a fast-scan STM experiment that is successfully imaging the rings, we must be cautious to use a low enough average current so that the tip also skates above the invisible C-H bonds that extend above the plane of the ring. Failure to do so would result in the tip constantly colliding with the aliphatic region of the molecule. The result would probably be a large amount of noise in the STM image, especially over the aliphatic regions. Indeed, this appears to be the case in some images of liquid crystals.[41]

We can also ask if it is even reasonable to image a single organic molecule on a graphite substrate in fast-scan mode. The possibility is not immediately obvious because we note that the absolute densities present in our images of the adsorbate are four or more orders of magnitude greater than the densities in the images of the bare surface. Therefore, can we establish an average sustainable tunneling current at a distance far enough above the graphite so that the tip also passes over the adsorbate? We will suggest here that this would be a difficult but plausible experiment.

First, note that our basis set was not optimized for examining the tails of wave functions as in our previous calculations because it was not necessary for generating most of the essential features of the imaging. At distances more than twice the normal C-C bond length, our graphite carbon basis functions fall off as $e^{-\alpha r^2}$ where α is the smallest basis set exponent. Of course, the proper behavior should be e^{-kr} where k is the vacuum decay constant for graphite. Therefore, the true offset in densities over the adsorbate vs. substrate can be a few orders of magnitude less due to the slower fall-off of the substrate wave function.

---

[41]W. Mizutani, M. Shigeno, M. Ono, K. Kajimura, *Appl. Phys. Lett.* **56** 1974 (1990).

Next, we note some interesting experiments that have been done with fast-scan images of porphyrin based molecules on a gold substrate by Luttrull *et al.*[42] In these experiments done in ambient conditions (room temperature, in air) isocyano groups attached to the porphyrin derivatives were used to chemically attach the adsorbates to the surface. Images could then be obtained (not atomic resolution) in fast-scan mode where the current fluctuations above the adsorbates were 100 times the average current set point of 1 nA. This demonstrates fast-scan imaging with at least two orders of magnitude in the current fluctuations. We also note that the Au substrate here will have a much slower decay into the vacuum than the graphite wave function.

From these two facts (our unaugmented basis set and the experimental evidence), we can suggest that single molecule fast-scan imaging on graphite might be possible with a sensitive instrument. Obvious suggestions to increase the chance of success of such an attempt would include chemically bonding the molecule to the substrate (difficult on graphite) and using cryogenic temperatures, as in the IBM Xe/Ni experiment.

---

[42]D. K. Luttrull, J. Graham, J. A. DeRose, D. Gust, T. A. Moore, S. M. Lindsay *Langmuir* **8** 765-768 (1992).

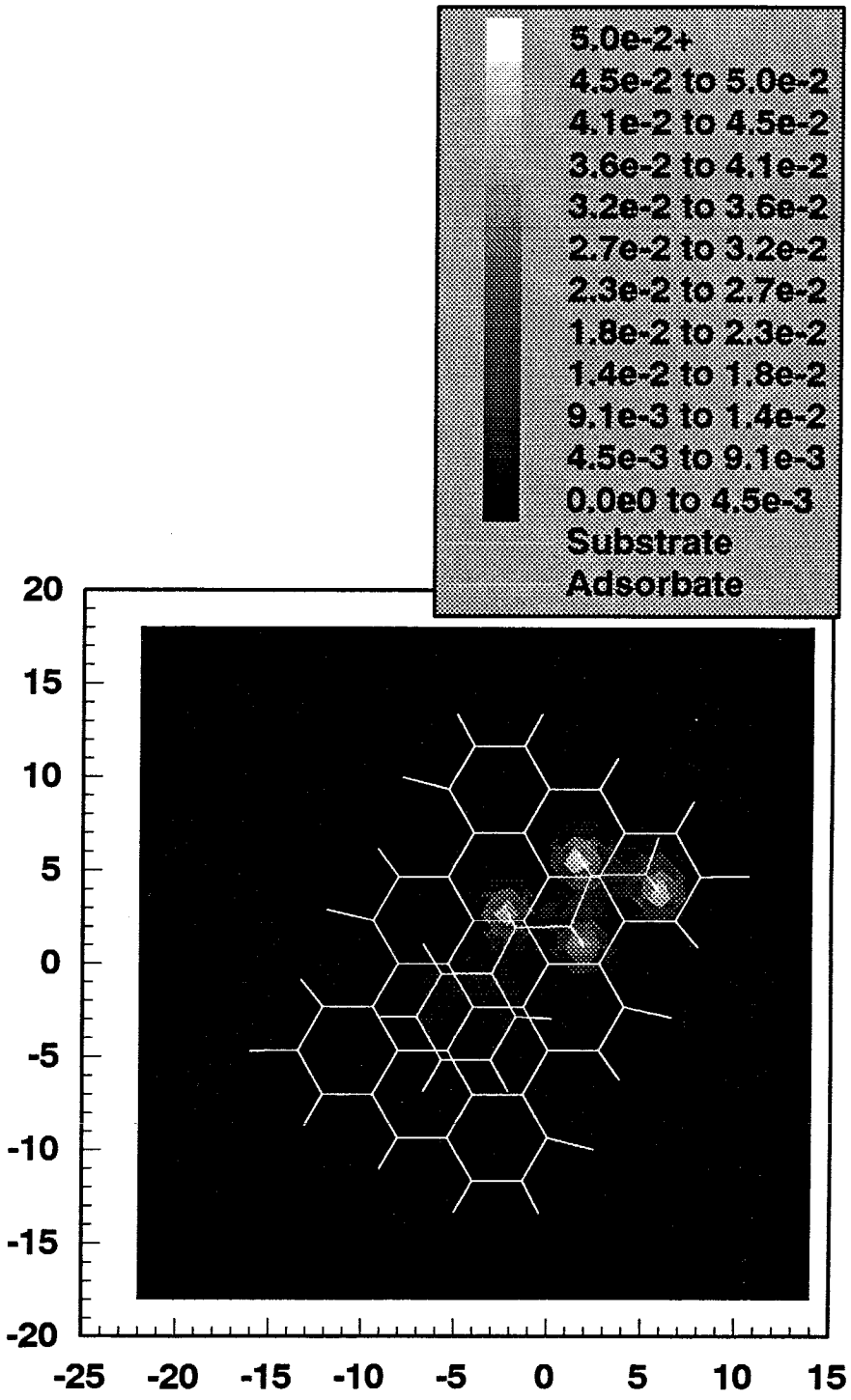


Figure 40: Density of states for all valence states (excludes O and C 1*s* orbital) shown with the molecular frame. Plot plane is z = -3.500 Å or 5.168 Å above the graphite and roughly 1.5 Å above the plane of the adsorbate molecule.
Contours are in units of bohr^{-3}.

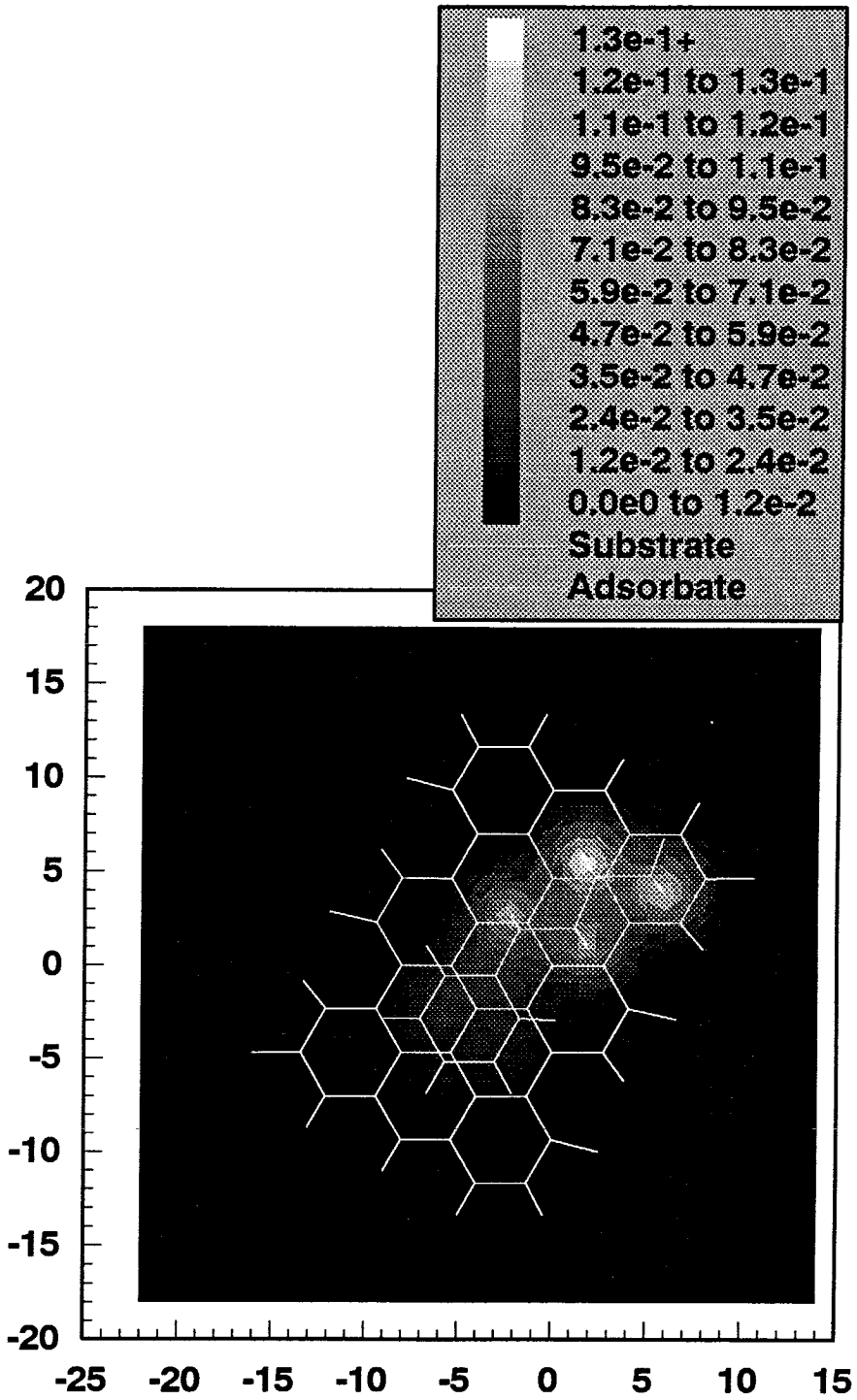


Figure 41:
Same as Figure 40 but with a 25-point filter applied. See text.

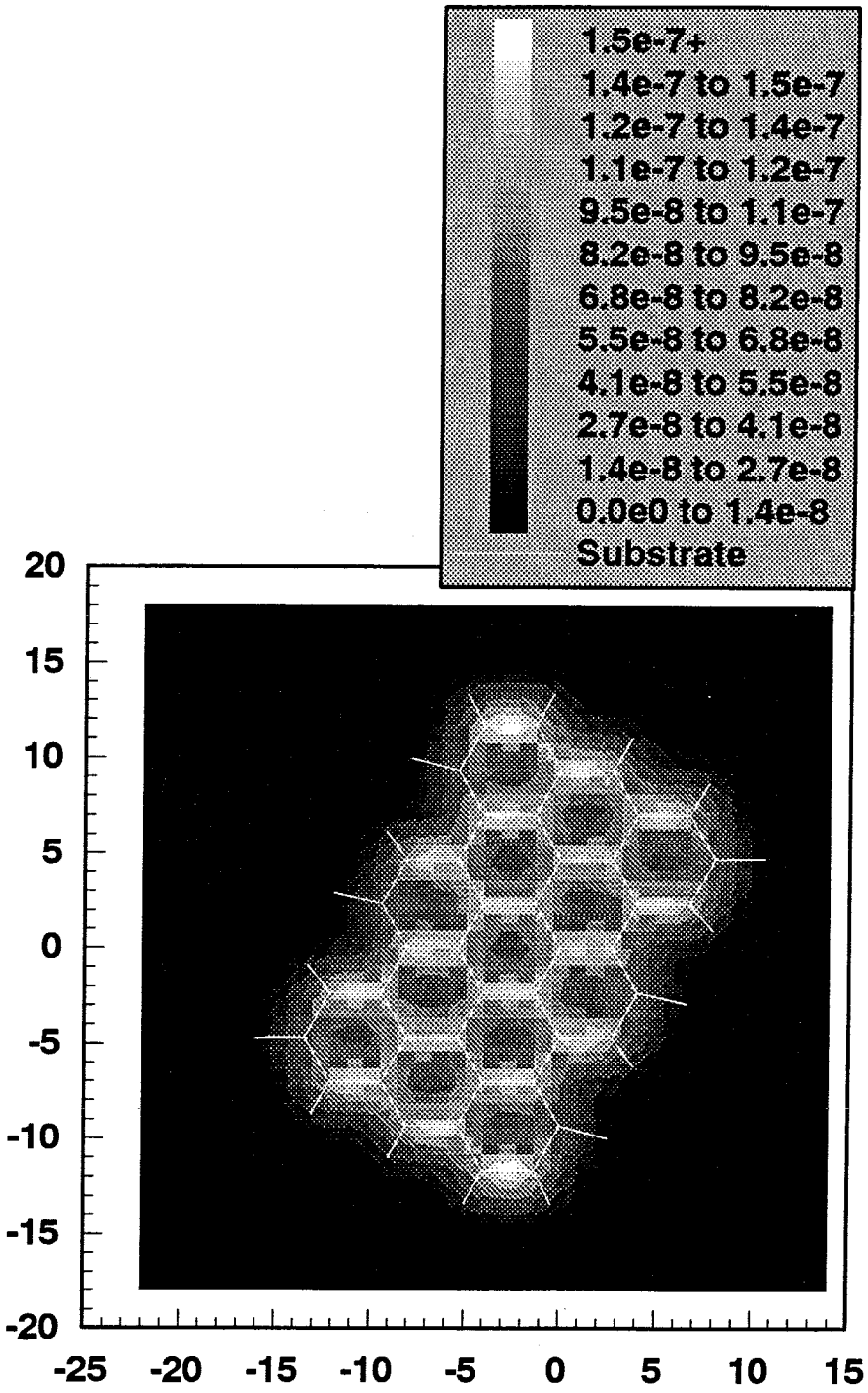


Figure 42:

Density of states for all valence states shown with the substrate framework. Plot plane is $z = 5.168$ Å or 3.5 Å from the graphite plane on the side *opposite* the adsorbate molecule. Contours are in units of bohr^{-3}.

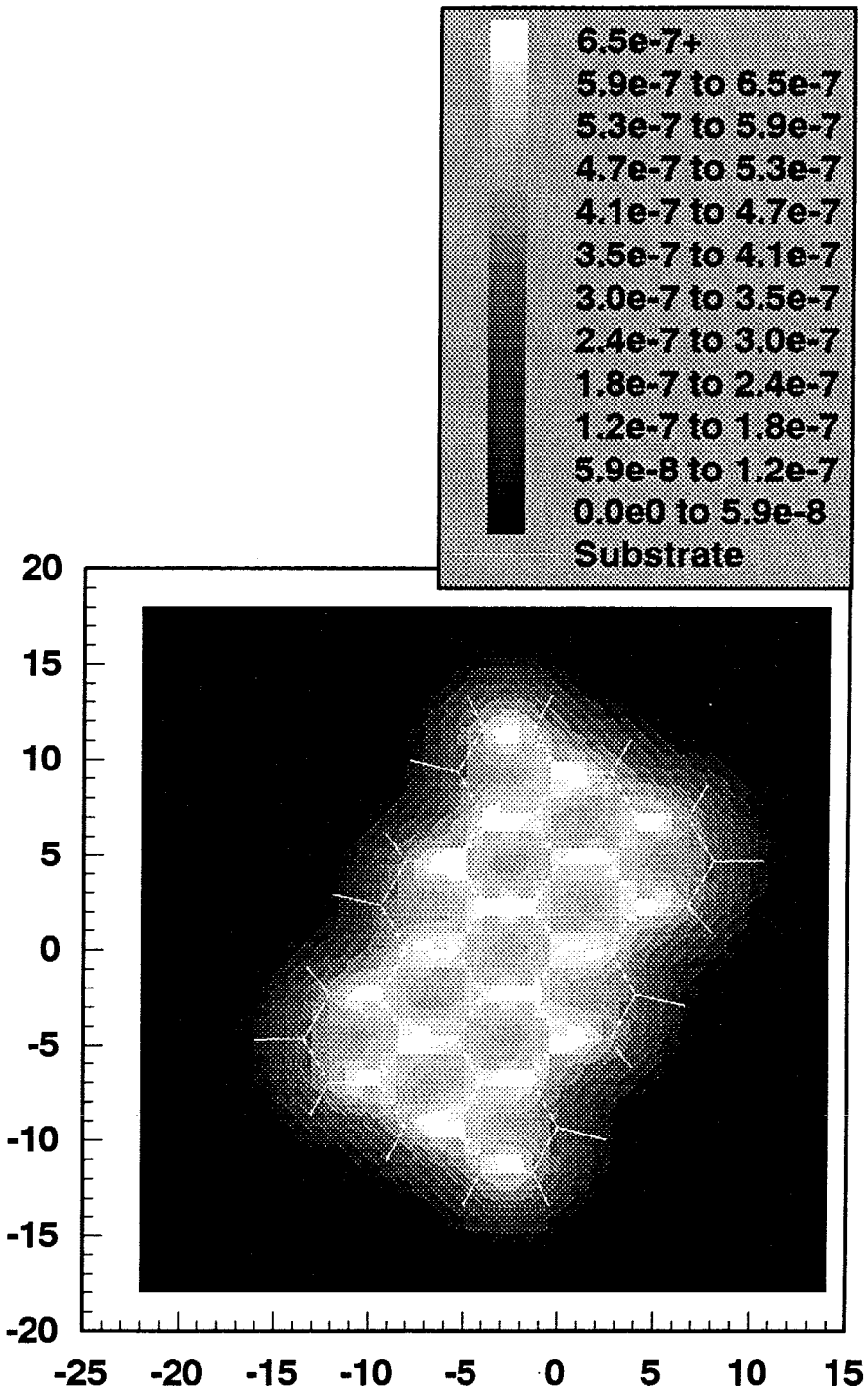


Figure 43:
Same as Figure 42 but with a 25-point filter applied. See text.

4.3.2. Topographic Images

SYNOPSIS:

We show total density isosurfaces in state energy windows near the Fermi energy. These isosurfaces are contrasted to the total density isosurfaces and correspond well with the types of images obtained by constant current STM. The STM current sensitivity required to image the adsorbate alkane chain is predicted to be roughly 50 times that needed for normal graphite imaging.

To predict constant current STM images we must find isosurfaces of the total density of states contributing to tunneling. In the Tersoff approximation, we expect the STM tip to trace these surfaces. In Figures 44 to 47 we have chosen the same energy windows with respect to the HOMO as in the fast-scan images. Two density isosurfaces are explored. The first at $5x10^{-4}$ electrons/bohr^3 generates an image consistent with the expected atomic resolution on the graphite surface. The second isosurface at $1x10^{-5}$ electrons/bohr^3 was chosen because it is the first density level consistent with imaging the alkane chain of the *n*-butyl benzene.

Again, the predicted images are relatively insensitive to the exact set of states chosen for imaging; the 4 eV window shows a slightly more smoothed graphite surface. For both the energy windows the states contain very little character around the alkane chain. As a result, we estimate the STM must be able to feedback to currents at least 50 times smaller than the currents used to achieve atomic resolution on graphite in order to allow the tip to pass over the alkane chain without collision.

Finally, it is useful to compare these images with an isosurface of constant *total* density. This highlights the differences between a true topological or AFM view of the

adsorbate/substrate system and the view of the STM experiment. A total density isosurface is shown in Figure 51.

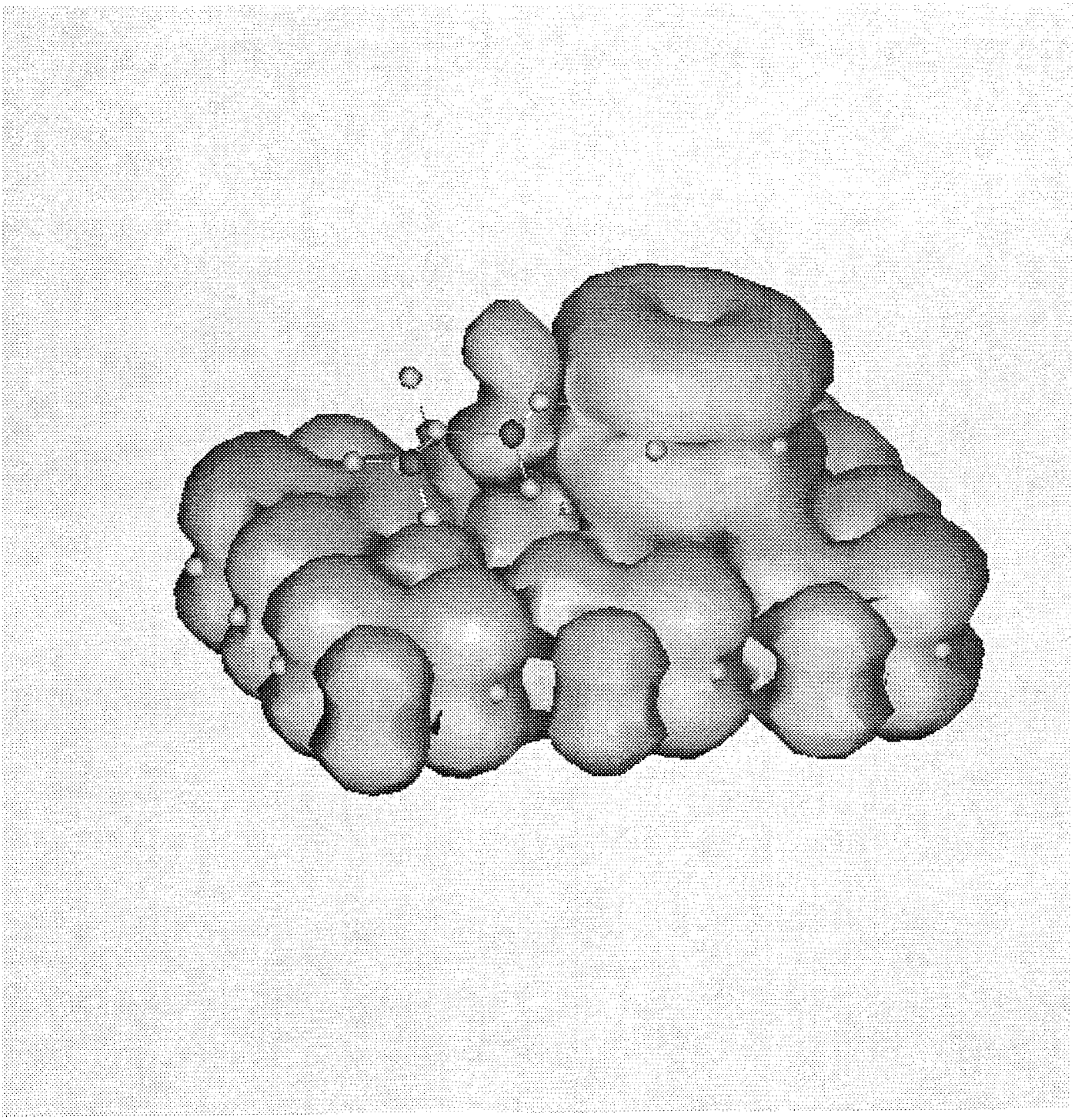


Figure 44:

Isodensity surface of 0.0005 electrons/bohr^3 for states with orbital energies from -0.34693 to -0.27281 hartree (a 2 eV window).

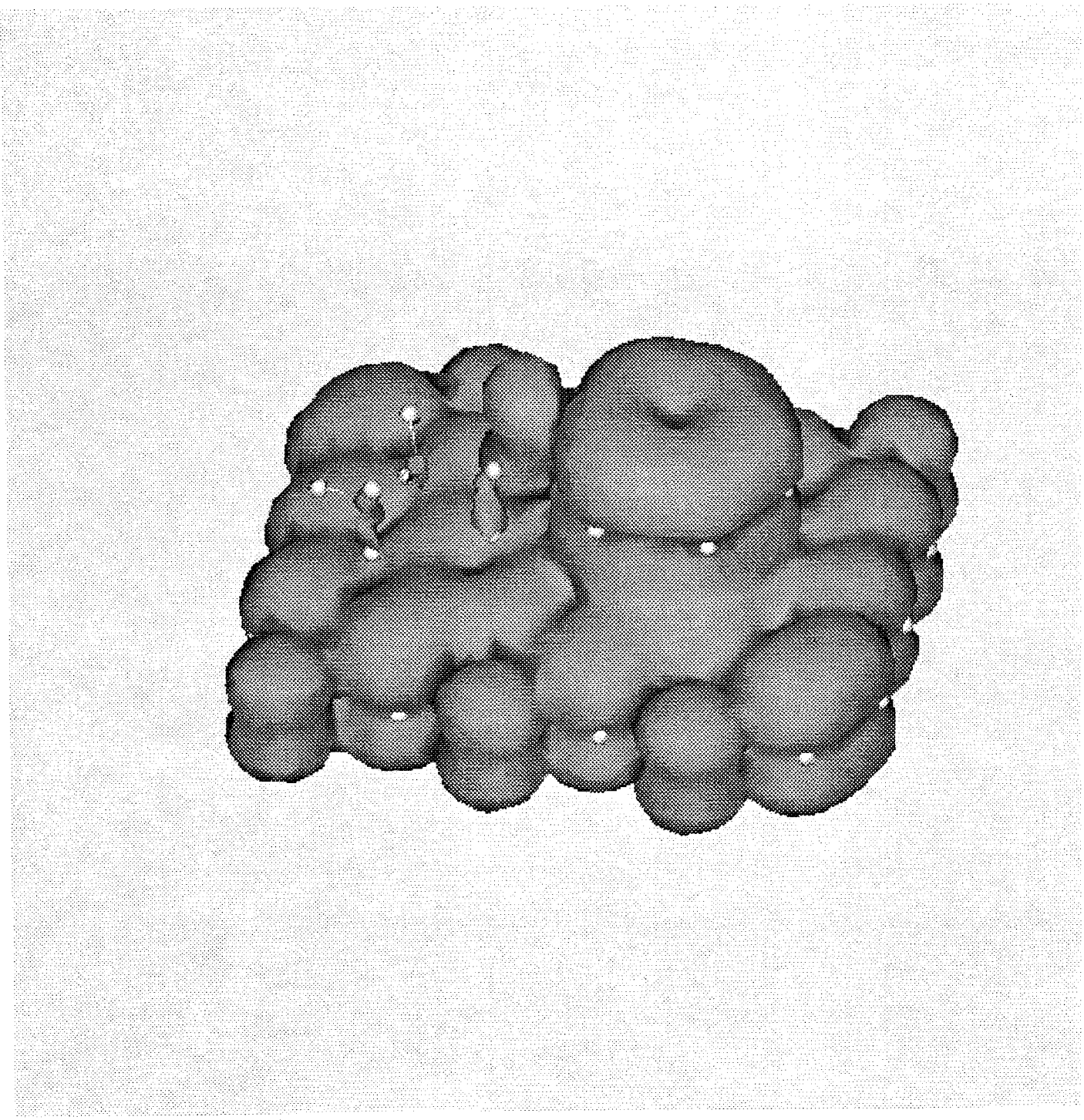


Figure 45:
Isodensity surface of 0.0005 electrons/bohr^3 for states with orbital energies from -0.41941 to -0.27281 hartree (a 4 eV window).

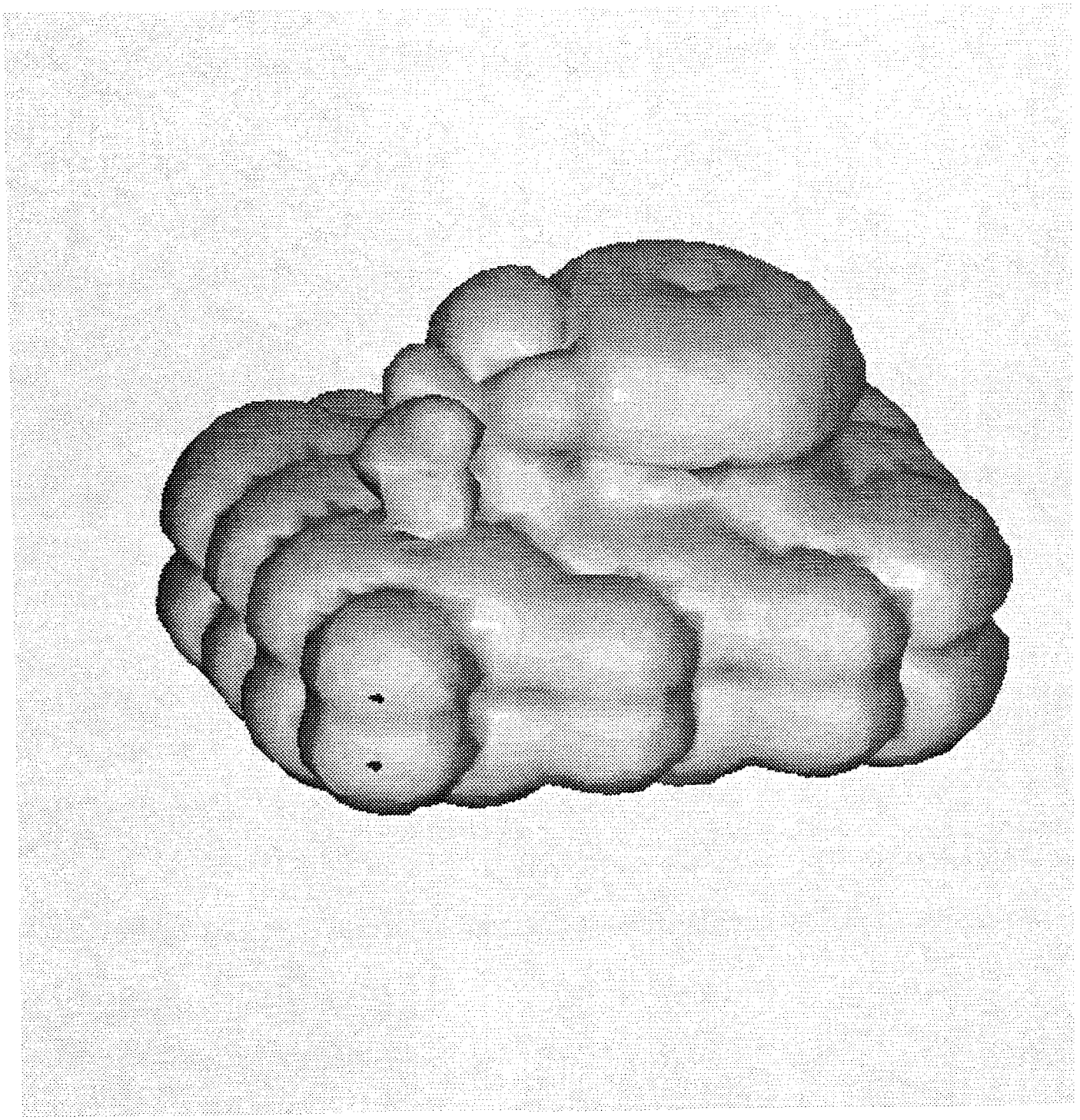


Figure 46:

Isodensity surface of 0.00001 electrons/bohr^3 for states with orbital energies from -0.34693 to -0.27281 hartree (a 2 eV window).

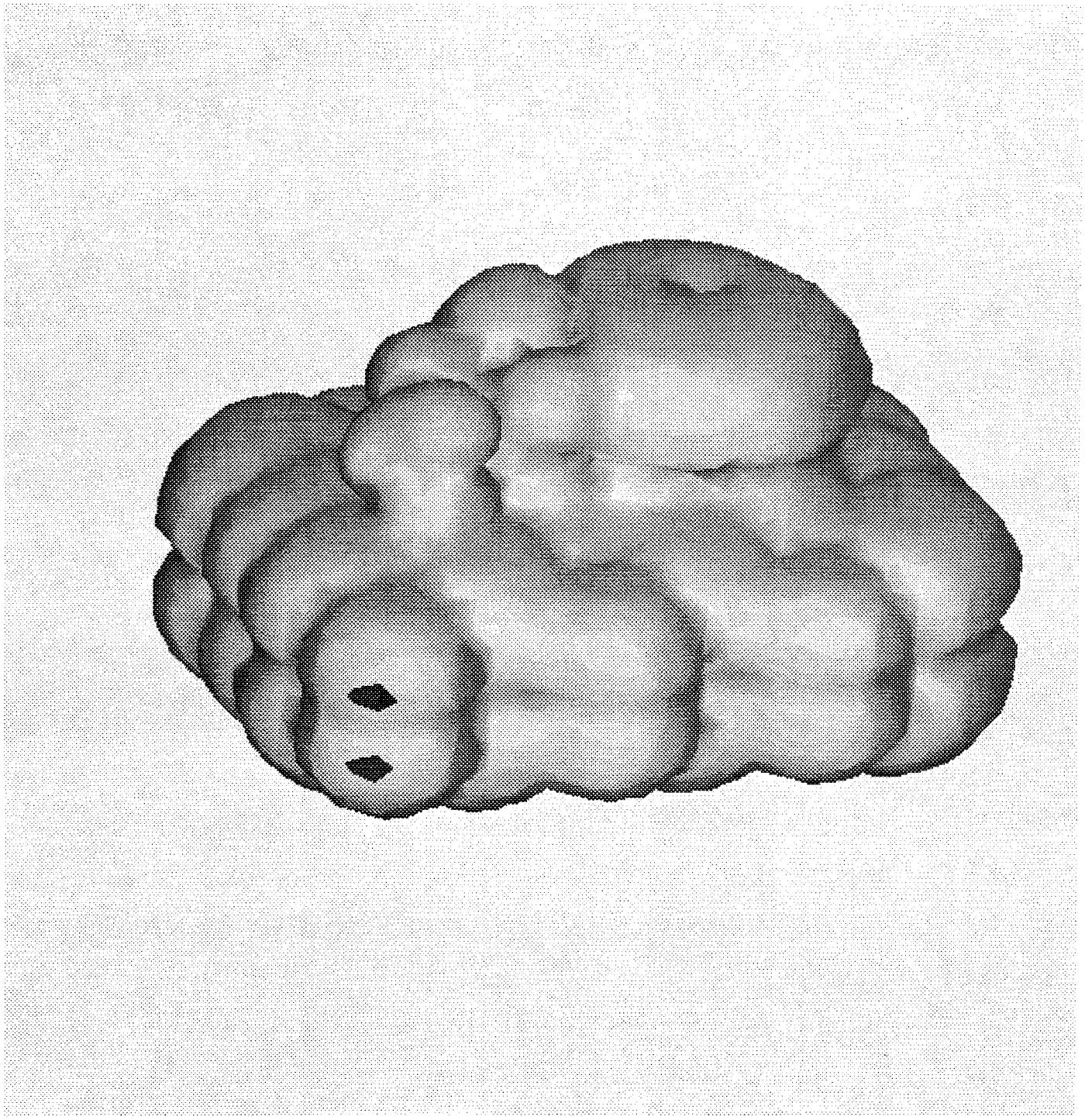


Figure 47:
Isodensity surface of 0.00001 electrons/bohr^3 for states with orbital energies from -0.41941 to -0.27281 hartree (a 4 eV window).

4.3.3. Barrier Height Images

SYNOPSIS:

Using the local ionization potentials of our model evaluated on surfaces of constant tunneling density we can predict STM barrier height images. The results show the potential for differentiating chemically distinct portions of adsorbates.

Local ionization potentials and barrier height imaging are defined in the thesis introduction and appendix, respectively. The barrier height imaging mode allows an additional piece of information to be collected simultaneously with the constant current STM contours. In our model we examine the same set of states and density isosurfaces discussed earlier but add the local ionization potentials to the tunneling density isosurfaces.

Figure 48 shows the 2 eV state window at 0.0005 electrons/bohr^3 isosurface color mapped with the local ionization potential. Red corresponds to the most easily ionized regions (dominated by the HOMO; pure red is a local ionization potential equal to the HOMO orbital energy) while blue corresponds to the most difficult to ionize regions (dominated by the lowest state of our energy window; blue is a local ionization potential equal to the orbital energy of the lowest state in the window). The most striking feature of this image is that the adsorbate ring appears as a dark spot (high barrier). This is in apparent contrast to the constant current or fast-scan images, both of which would image the ring as a bright spot (high current). In the constant current mode, the ring would be bright due to the measured tip retraction over the adsorbate; in the fast-scan image the ring is bright due to the greater spatial extent into the vacuum of the ring wave function. The reason that the *n*-butyl benzene ring appears dark is that the state density associated with this ring is prevalent in the states a few levels below the HOMO of the

graphite/adsorbate system. Rather, graphite character dominates the highest few states. This becomes apparent upon examination of the seven states contributing to the tunneling density we are plotting. We have not plotted all seven here, but see Figure 33 for one example. This state, three down from the HOMO, does contribute appreciable density to the ring.

These results are consistent with experimental results obtained by Driscoll and Youngquist[43] while tunneling through DNA to graphite. In these experiments, they found regions where higher barrier heights corresponded with higher topography and vise versa. The two regions of different behavior corresponded to the backbone and base pair regions of the DNA molecule. The resolution is not such that a definite chemical association can be made, however, it is clear that the regions are chemically distinct.

Note that no contradiction between the barrier height information and the tunneling current predictions is present, even though at first glance one might always expect higher currents at the lowest barrier heights. The variable unaccounted for in this analysis is the tip-sample distance. This distance is free to vary in the constant current experiment, and probably does. Therefore, if the tip has retracted to image over a protruding adsorbate, the STM will image a topological bright spot. However, the *relative* distance from tip to sample (adsorbate) may have decreased (increased) due to an increased (decreased) local barrier height.

In this calculation our basis set is not flexible enough to resolve these issues of relative tip-sample separation as a function of local barrier height by examining the tail of the wave function. As we have found through careful study of smaller systems (Xe/Ni), we would need to add diffuse functions to describe the vacuum decay with more precision.

---

[43]R. J. Driscoll, M. J. Youngquist, personal communication. See also the respective theses, California Institute of Technology, 1992 and 1993 respectively.

Enough degrees of freedom would be needed to ensure that higher energy states could use the diffuse functions in varying combinations and amounts to reproduce the vacuum decay rates. The number of functions needed would be prohibitive at this point in time. However, the state energies provide a convenient substitute for this information. Because the relative energies of the cluster model states are well described, we can use the orbital energies instead of calculating vacuum tail decay constants to get relative barrier heights. This is also corroborated by our findings in the Xe/Ni cases that the ratios of barrier heights computed by fitting the exponential tails were similar to those expected from work function ratios.

Finally, we note that there is also barrier height variance on the graphite surface itself. This is due primarily to the finite size and chemical inhomogeneity of the cluster model. For the set of states imaged in Figures 48 to 50, however, the graphite barrier height is always lower than the adsorbate.

Although the STM always images in projection, we are not limited to this on the computer. Figure 49 gives an edge view of the surface traced by the STM and the local barrier height expected to be imaged by these states. We have also explored the more diffuse 0.00001 electrons/bohr^3 surface corresponding to an instrument 50 times more sensitive. The view in Figure 50 a) shows how close the constant density surface is to the alkane chain even at this small value, and b) demonstrates an increased barrier height over the alkane chain. That is, only the lowest states of the set are contributing any density at all to the adsorbate alkane chain.

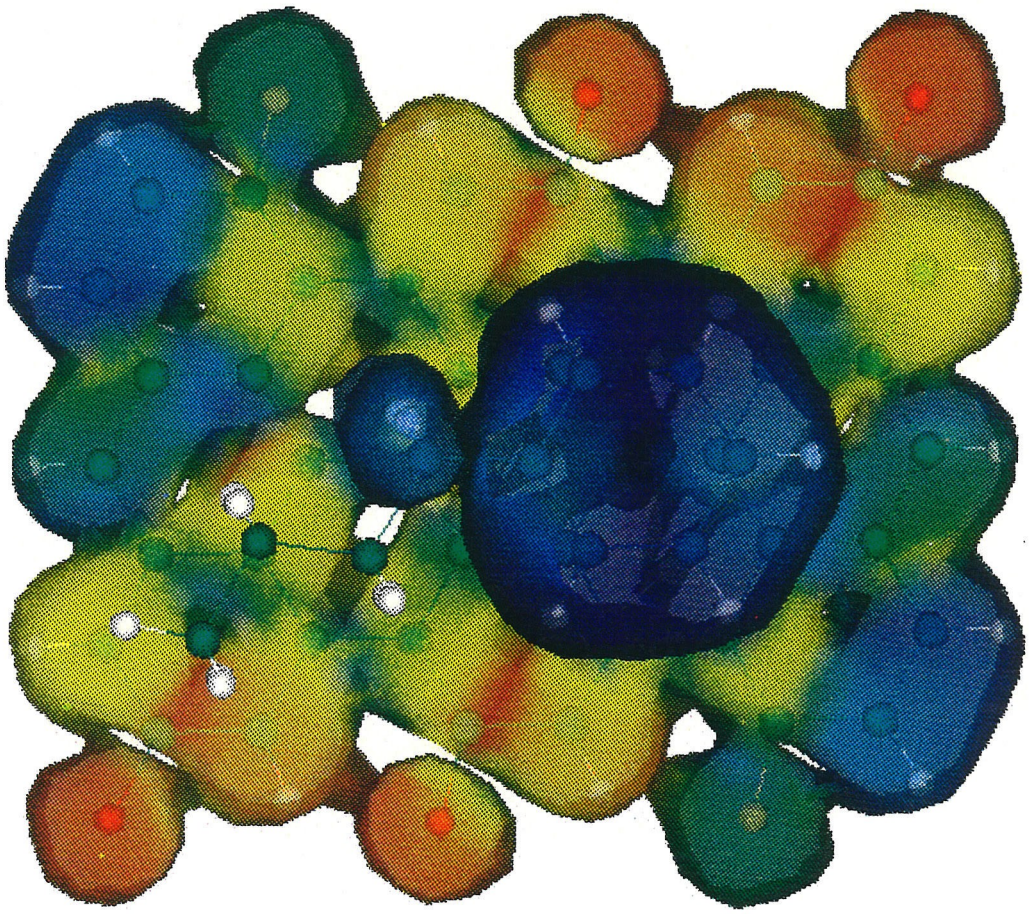


Figure 48: Isodensity surface of 0.0005 electrons/bohr^3 for states with orbital energies from -0.34693 to -0.27281 hartree (a 2 eV window) color mapped with the local ionization potential. Red corresponds to the lowest barrier regions (most easily ionized) and blue corresponds to the highest local barriers.

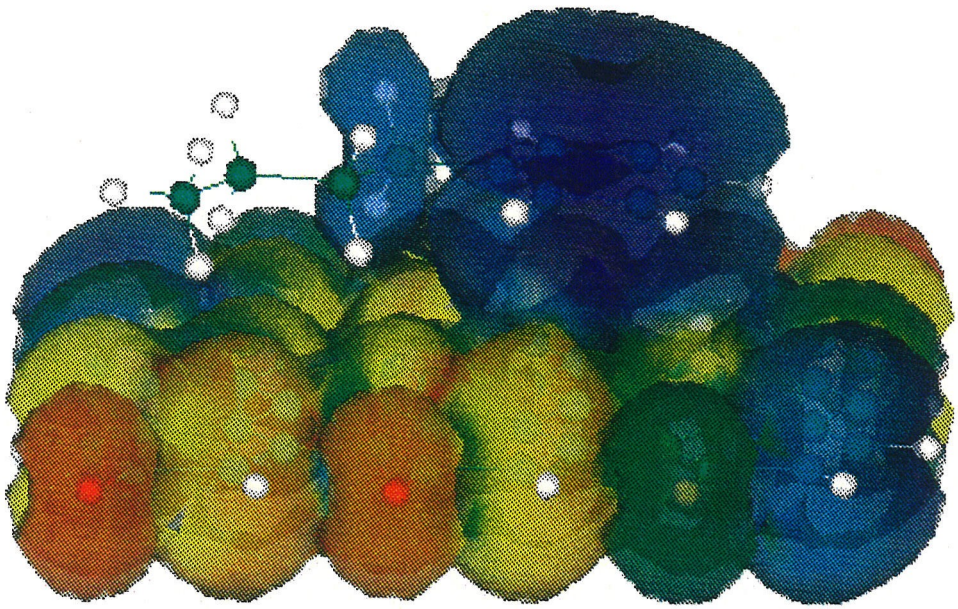


Figure 49:
Edge view of Figure 48.

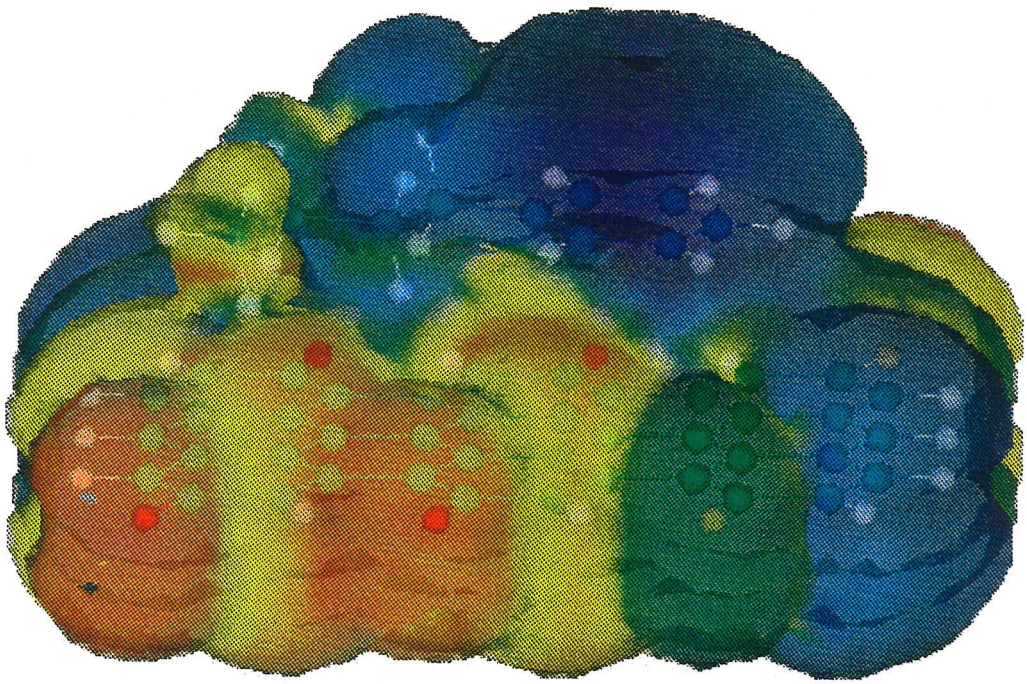


Figure 50:
Similar to 48 but with isosurface of 0.00001 electrons/bohr^3 in order to show increased barrier over the alkane chain.

4.3.4. AFM Images

SYNOPSIS:

Plots of total electron density show the images expected from an ideal AFM experiment.

Our calculations allow us to predict the result of an ideal atomic force microscope (AFM) experiment. The repulsive force between a probe and the sample will approximately follow contours of total electron density. Figure 51 shows a total electron density isosurface of 0.001 electrons/bohr^3 computed for all valence states. (The core O and C 1*s*-states do not contribute significant density anywhere on this surface since they are much closer to the nucleus and die off rapidly.)

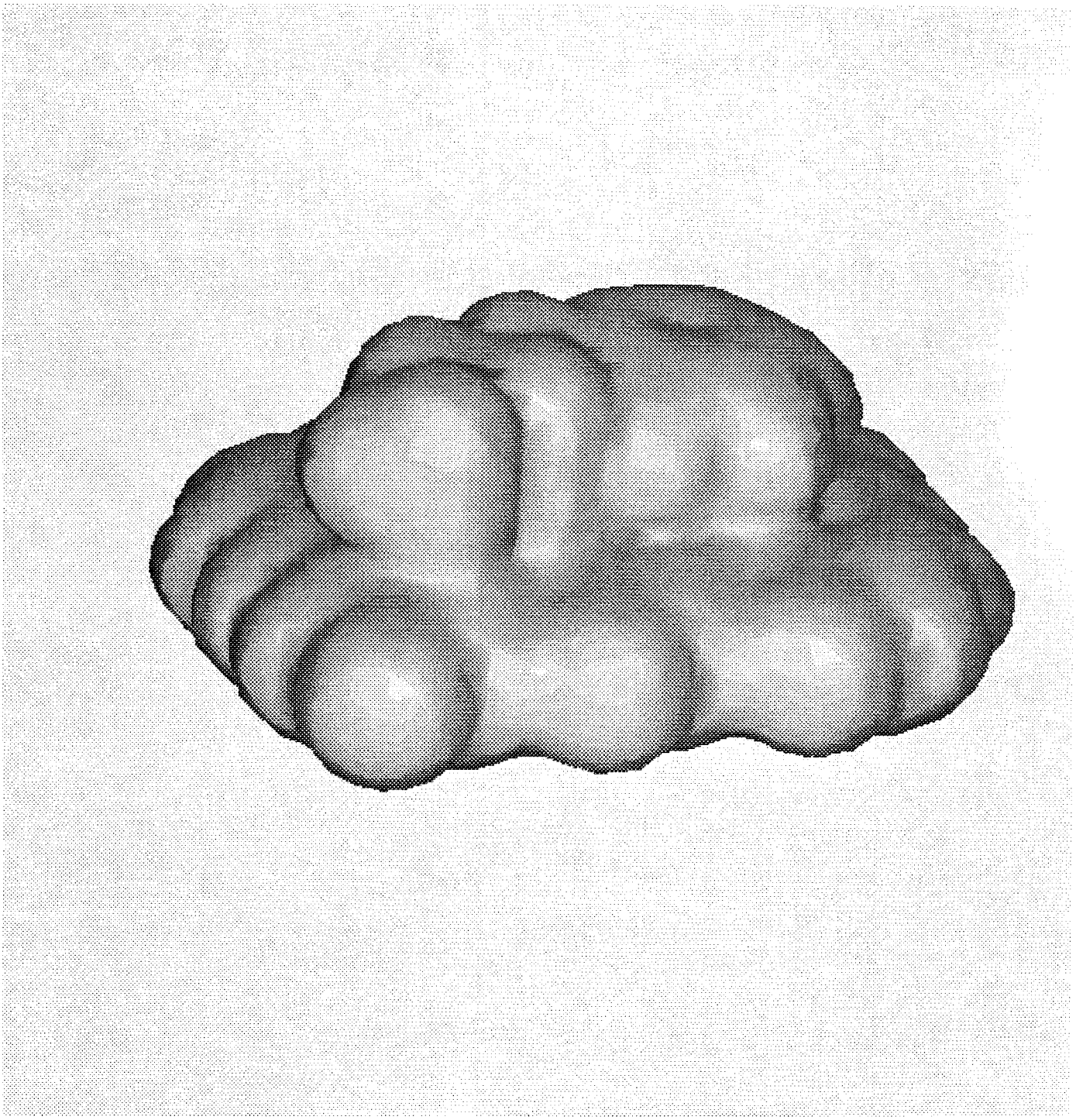


Figure 51:
Isodensity surface of 0.001 electrons/bohr^3 for all valence states.

4.4. Discussion

In Figure 52 we have reproduced an experimental image of graphite.[44] This image was taken at 8 mV tip bias, 1 nA current, and has a corrugation of roughly 1 Å. It was taken under ultra-high vacuum with an Ir tip. The image lacks any signs of double tip effects and is therefore a good candidate for comparison with our images. It may also be compared with the results of the next chapter.

Next we comment briefly on some other experimental literature and how it relates to our results. Mizutani *et al.*[41] have imaged 8CB (4-octyl-4'-cyanobiphenyl) on graphite at 1.0 V and 0.1 V biases and found that they image the adsorbate or the underlying graphite, respectively. This is consistent with our findings. When the graphite is being imaged, the results are noisy. This is probably due to perturbations of the graphite by the adsorbate. When the adsorbate is imaged, resolution again was not very good, but they appear to image the biphenyl, again consistent with the invisibility of the alkane chain.

McGonigal *et al.*[45] have done some excellent imaging of packed arrays of n-$C_{17}H_{36}$ on graphite. They find that they are able to see lines of packed alkanes with the underlying graphite substrate visible (with some filtering). The imaging mechanism here is perturbation of the graphite substrate wave function by the adsorbate (as in the Xe/Ni case). States of the alkane are not being imaged directly. The voltage was -0.5 V tip bias at 0.5 nA.

Lippel *et al.*[46] imaged Cu phthalocyanine on Cu(100) and compared the images to the phthalocyanine HOMO from extended Hückel calculations. The results were

---

[44]J. K. Gimzewski, E. Stoll, R. R. Schlittler, *Surf. Sci.* **181** 267-277 (1987).

[45]G. C. McGonigal, R. H. Bernhardt, D. J. Thomson, *Appl. Phys. Lett.* **57** 28 (1990).

[46]P. H. Lippel, R. J. Wilson, M. D. Miller, Ch. Wöll, S. Chiang, *Phys. Rev. Lett.* **62** 171 (1989).

reasonable, however, they simply compared one orbital (the HOMO) with the STM image. The single orbital necessarily had localized lobes that did not exist in the STM image. This analysis is not consistent with their 2 V bias voltages - summing density over contributing states would yield a better prediction. Also, their adsorbate was not calculated in the presence of the substrate, although this is probably not critical when imaging relatively localized states of the adsorbate itself.

Spong *et al.*[47] imaged various liquid crystal on graphite systems. They report barrier heights of 0.06 eV to 0.10 eV. It seems clear that the tunneling gap has broken down and that they are imaging in contact. We have not addressed mechanisms for imaging in this regime; forces on the tip come into play as well as tunneling currents. Lang[48] has predicted the breakdown of the tunneling barrier for very short distances. Spong *et al.* discuss a mechanism for contrast by suggesting the polarizable adsorbate modulates the barrier of the substrate. Their explanation does not contain chemical intuition about the electronic states contributing to tunneling. In any case, it would be easier to assess these outlooks if their experiments were run without apparently collapsing the tunneling barrier.

Finally, the interested reader may also wish to examine the work of Hallmark *et al.*[49] who imaged naphthalene on Pt (111).

---

[47]J. K. Spong, H. A. Mizes, L. J. LaComb Jr, M. M. Dovek, J. E. Frommer, J. S. Foster, *Nature* **338** 137 (1989).

[48]N. D. Lang, *Comments Cond. Mat. Phys.* **14** 253-275 (1989).

[49]V. M. Hallmark, S. Chiang, J. K. Brown, Ch. Wöll, *Phys. Rev. Lett.* **66** 48 (1991).

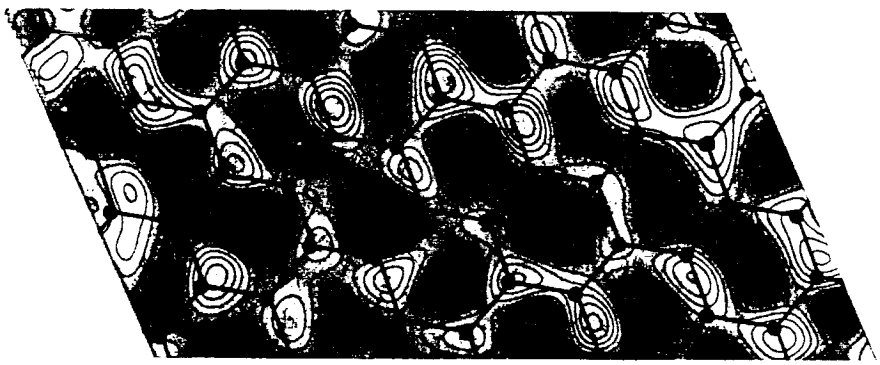


Fig. 1. STM top view of a graphite surface imaged with an Ir tip at $V_t = 8$ mV, $i_t = 1$ nA. Grey-tone scale range is ~1 A. Darker areas represent lower heights.

Figure 52:

Figure from Ref. (44) of graphite under UHV, Ir tip at 8 mV, 1 nA current. Corrugation is roughly 1 Å.

4.5. Appendix

The basis sets used below are the standard 6-31G basis sets. The parameters are reproduced here for convenience:

C90DZ:

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
S/1	0.30475249D+04	0.18347371D-02
S/1	0.45736952D+03	0.14037323D-01
S/1	0.10394868D+03	0.68842622D-01
S/1	0.29210155D+02	0.23218444D+00
S/1	0.92866630D+01	0.46794135D+00
S/1	0.31639270D+01	0.36231199D+00
S/2	0.78682723D+01	-0.11933242D+00
S/2	0.18812885D+01	-0.16085415D+00
S/2	0.54424926D+00	0.11434564D+01
P/1	0.78682723D+01	0.68999067D-01
P/1	0.18812885D+01	0.31642396D+00
P/1	0.54424926D+00	0.74430829D+00
S/3	0.16871448D+00	1.00000000D+00
P/2	0.16871448D+00	1.00000000D+00

O90DZ:

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
S/1	0.54846717D+04	0.18310744D-02
S/1	0.82523495D+03	0.13950172D-01
S/1	0.18804696D+03	0.68445078D-01
S/1	0.52964500D+02	0.23271434D+00
S/1	0.16897570D+02	0.47019290D+00
S/1	0.57996353D+01	0.35852085D+00
S/2	0.15539616D+02	-0.11077755D+00
S/2	0.35999336D+01	-0.14802626D+00
S/2	0.10137618D+01	0.11307670D+01
P/1	0.15539616D+02	0.70874268D-01
P/1	0.35999336D+01	0.33975284D+00
P/1	0.10137618D+01	0.72715858D+00
S/3	0.27000582D+00	1.00000000D+00
P/2	0.27000582D+00	1.00000000D+00

H90DZ:

Function type/ Sequence number	Radial exponent α_i	Contraction coefficient k_i
S	0.18731137D+02	0.33494604D-01
S	0.28253944D+01	0.23472695D+00
S	0.64012169D+00	0.81375733D+00
S	0.16127776D+00	1.00000000D+00

Following are the coordinates used for the *n*-butyl·benzene on graphite model:

Label	Basis Set	Element	X (Å)	Y (Å)	Z (Å)
C1	C90DZ	C	0.00000	0.00000	1.66803
C2	C90DZ	C	1.42514	-0.00001	1.66803
C3	C90DZ	C	-4.98795	-1.23421	1.66803
C4	C90DZ	C	-6.41295	-1.23422	1.66803
C5	C90DZ	C	-7.12544	-2.46832	1.66803
C6	C90DZ	C	-4.27539	-2.46840	1.66803
C7	C90DZ	C	-4.98789	-3.70249	1.66803
C8	C90DZ	C	-6.41289	-3.70249	1.66803
C9	C90DZ	C	-2.85025	-2.46841	1.66803
C10	C90DZ	C	-4.27539	0.00000	1.66803
C11	C90DZ	C	-4.98790	1.23408	1.66803
C12	C90DZ	C	-2.85025	-0.00001	1.66803
C13	C90DZ	C	-2.85025	2.46839	1.66803
C14	C90DZ	C	-4.27525	2.46838	1.66803
C15	C90DZ	C	-2.13770	-3.70260	1.66803
C16	C90DZ	C	-2.85021	-4.93668	1.66803
C17	C90DZ	C	-4.27521	-4.93666	1.66803
C18	C90DZ	C	-0.71256	-3.70261	1.66803
C19	C90DZ	C	-2.13770	-1.23420	1.66803
C20	C90DZ	C	-0.71256	-1.23421	1.66803
C21	C90DZ	C	-2.13770	1.23420	1.66803
C22	C90DZ	C	-0.71256	1.23419	1.66803
C23	C90DZ	C	-2.13770	3.70260	1.66803
C24	C90DZ	C	-2.85021	4.93668	1.66803
C25	C90DZ	C	-0.71256	3.70259	1.66803
C26	C90DZ	C	-0.71256	6.17099	1.66803
C27	C90DZ	C	-2.13756	6.17098	1.66803
C28	C90DZ	C	0.00000	-4.93680	1.66803
C29	C90DZ	C	-0.71250	-6.17089	1.66803
C30	C90DZ	C	-2.13750	-6.17089	1.66803
C31	C90DZ	C	0.00000	-2.46840	1.66803
C32	C90DZ	C	1.42514	-2.46841	1.66803
C33	C90DZ	C	0.00000	2.46840	1.66803
C34	C90DZ	C	1.42514	2.46839	1.66803

C35	C90DZ	C	0.00000	4.93680	1.66803
C36	C90DZ	C	1.42514	4.93679	1.66803
C37	C90DZ	C	2.13770	-1.23420	1.66803
C38	C90DZ	C	2.13770	1.23420	1.66803
C39	C90DZ	C	3.56284	1.23419	1.66803
C40	C90DZ	C	2.13770	3.70260	1.66803
C41	C90DZ	C	3.56284	3.70259	1.66803
C42	C90DZ	C	4.27539	2.46840	1.66803
H43	H90DZ	H	-7.04860	-0.46687	1.66803
H44	H90DZ	H	-6.93623	-4.58204	1.66803
H45	H90DZ	H	-4.86748	3.27041	1.66803
H46	H90DZ	H	-4.78737	-5.82393	1.66803
H47	H90DZ	H	-0.21444	7.06507	1.66803
H48	H90DZ	H	-2.61937	7.07674	1.66803
H49	H90DZ	H	-0.23070	-7.07665	1.66803
H50	H90DZ	H	-2.63559	-7.06498	1.66803
H51	H90DZ	H	2.01735	-3.27035	1.66803
H52	H90DZ	H	1.93737	5.82402	1.66803
H53	H90DZ	H	4.19849	0.46689	1.66803
H54	H90DZ	H	4.08617	4.58212	1.66803
O55	O90DZ	O	-6.30961	1.53261	1.66803
O56	O90DZ	O	-4.15886	5.27992	1.66803
O57	O90DZ	O	-8.47897	-2.48632	1.66803
O58	O90DZ	O	1.30865	-5.27994	1.66803
O59	O90DZ	O	3.45942	-1.53265	1.66803
O60	O90DZ	O	5.62889	2.48641	1.66803
C61	C90DZ	C	-3.02460	-0.30844	-1.92300
H62	H90DZ	H	-3.55594	0.56479	-1.97607
C63	C90DZ	C	-1.60762	-0.28581	-1.94651
C64	C90DZ	C	-0.92269	-1.52560	-1.86291
H65	H90DZ	H	0.09709	-1.56751	-1.87000
C66	C90DZ	C	-1.62966	-2.73913	-1.76958
H67	H90DZ	H	-1.12089	-3.62427	-1.71061
C68	C90DZ	C	-3.03451	-2.73743	-1.75577
H69	H90DZ	H	-3.54764	-3.61937	-1.68846
C70	C90DZ	C	-3.73264	-1.52066	-1.82948
H71	H90DZ	H	-4.75512	-1.51649	-1.81522
C72	C90DZ	C	-0.93188	1.04234	-2.06531
H73	H90DZ	H	-1.28173	1.50822	-2.99080
H74	H90DZ	H	-1.27391	1.66447	-1.23474
C75	C90DZ	C	0.62279	1.07436	-2.07962
H76	H90DZ	H	0.98266	0.50667	-2.94250
H77	H90DZ	H	0.99910	0.59745	-1.17249
C78	C90DZ	C	1.19551	2.51522	-2.15555
H79	H90DZ	H	0.83403	3.09690	-1.30404
H80	H90DZ	H	0.84092	2.99861	-3.06982
C81	C90DZ	C	2.74049	2.54614	-2.15126
H82	H90DZ	H	3.13432	2.02087	-3.02369
H83	H90DZ	H	3.08933	3.58010	-2.18001
H84	H90DZ	H	3.13040	2.07325	-1.24900

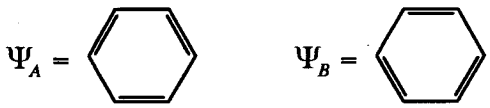
Table 8: *n*-butyl benzene on graphite model.

5. New Computational Approaches to Tunneling

5.1. Bardeen's Tunneling Matrix Elements from *ab initio* Wave Functions

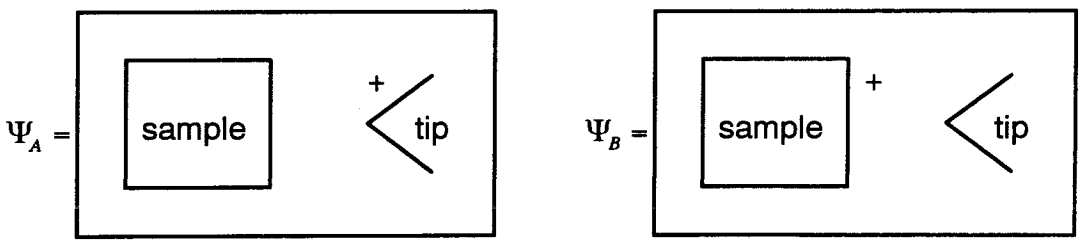
5.1.1. Introduction

The methods developed in Ref. (8) were originally targeted toward evaluating chemical resonance energies using *ab initio* Hartree-Fock and GVB wave functions. The prototypical example is benzene where



The resonance wave function, $\Psi_{RES} = \Psi_A + \Psi_B$, captures the most important reference states suggested by chemical intuition. The energy of the resonance wave function recovers a large portion of the energy missed in the single reference description.

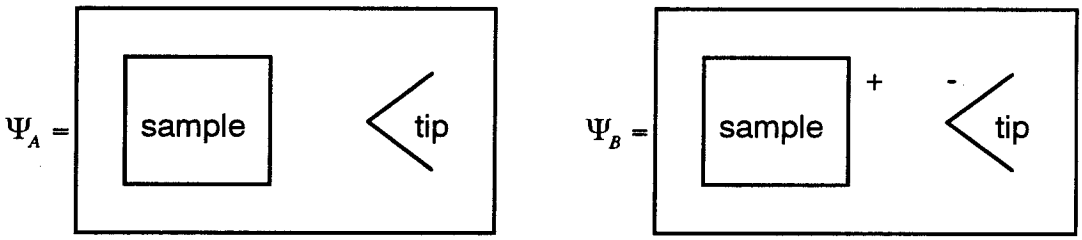
Following some electron transfer work by Cave *et al.*,[50] we have utilized the same machinery to describe a different type of system, namely the (two state) electron transfer complex where



[50]R. J. Cave, Ph.D. Thesis, California Institute of Technology, (1986); P. West, J. Kramar, D. V. Baxter, R. J. Cave, J. D. Baldeschwieler, *IBM J. Res. Develop.* **30** 484 (1986).

The same formalism may be used because the same conditions hold for the two wave functions as for the resonance case. Specifically, each state has the same number of electrons and the same orbital occupation numbers. Only the distribution of charge has changed. The matrix elements between these states incorporate the rearrangement of all N electrons in response to the electron transfer.

There are, however, some drawbacks to using this description for the electron transfer description in the context of *ab initio* calculations. The largest drawback is the difficulty of converging good descriptions of both states. Without very careful design of the electron transfer model, the two states are generally sufficiently different in energy that it becomes difficult to converge the high energy state. Another difficulty is the necessity of using a charged cluster to describe the electron transfer. The reason for this becomes apparent upon examining how a neutral cluster model would work:



At least one of the states in this system would need to be a charge-separated species. While this may be a useful description for some systems, e.g., photo-induced charge transfer, it is unlikely this is the best description for the STM case. Convergence of the charge separated species presents additional technical difficulties.

Finally, it is both a strength and a weakness of the method that the two electrodes (we refer to the tip and the sample) are described in the same composite wave function. In principle, this provides an excellent many-body description of the electronic rearrangement that occurs upon electron transfer. However, for the small model systems that we must

use, this results in a significantly larger polarization of the electrodes than probably occurs in the actual bulk systems. The degree to which this is a problem depends on the system being studied. If electron transfer to localized adsorbate states is being studied, this description is probably good. If we are studying transfer into delocalized bulk states, the error is greater.

5.1.2. The Bardeen Transfer Matrix Element

Here we develop a method for computing the Bardeen[51] transfer matrix element using *ab initio* Hartree-Fock or GVB wave functions. We will see that the use of a Gaussian basis set makes it particularly easy to compute the matrix elements in Bardeen's perturbation approach to electron transfer. Bardeen originally solved for the transfer matrix elements to describe tunneling in planar metal-oxide-metal junctions. The work has been expanded upon by others to describe the STM tunneling problem.[52,53,54]

5.1.2.1. Tunneling Current

The tunneling current from sample to tip in first-order perturbation theory is:[52]

$$I = \left(\frac{2\pi e}{\hbar}\right)\sum_{st}|T_{st}|^2 f(E_s)[1 - f(E_t + eV)]\delta(E_s - E_t) \qquad (5.1)$$

where the sum is over all sample states, s, and tip states, t; $f(E)$ is the Fermi function; V is the bias voltage; E is the energy of the indexed (unperturbed) state of the sample or tip for infinite sample-tip separation. The Fermi function provides occupation fractions for the sample and tip states (excludes tunneling into occupied states) while the delta function

---

[51]J. Bardeen, *Phys. Rev. Lett.* **6** 57 (1961).

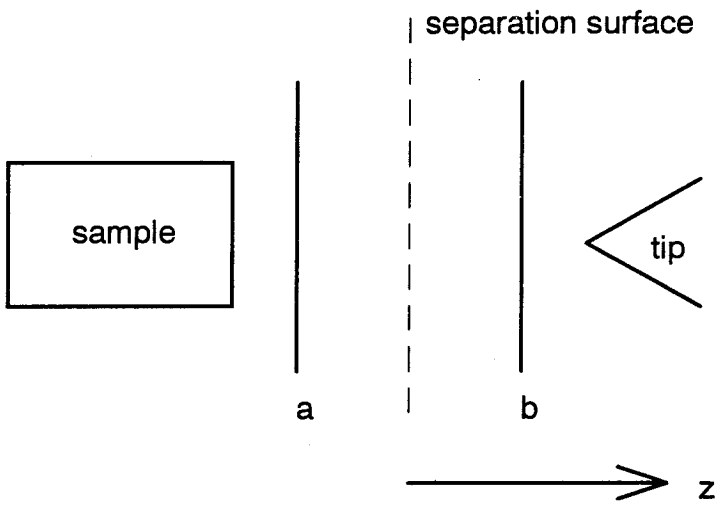
[52]J. Tersoff, D. R. Hamann, *Phys. Rev. Lett.* **50** 1998 (1983).

[53]J. Tersoff, *Phys. Rev. B* **40** 11990 (1989).

[54]C. J. Chen, *J. Vac. Sci. Tech. A* **6** 319 (1988).

specifies that the contributions to tunneling occur only between states of equal energy in the sample and tip. T_{st} is the tunneling matrix element $\langle \varphi_s | W | \varphi_t \rangle$ where W is the perturbing part of the Hamiltonian present because of the finite separation between sample and tip. Bardeen's contribution was to show that the matrix elements can be calculated under certain circumstances purely from a knowledge of the wave function (sample and tip states independently) and no information on the form of W.

By specifically using the geometry of the tunneling problem and the unperturbed eigenstates of the sample and tip, Bardeen expressed the tunneling matrix elements as surface integrals over a region lying entirely between the two electrodes. The integrand contains only a current operator and the states of the unperturbed system.



The following assumptions are made: $\{\varphi_s\}$, the states of the sample, are well described in the range $-\infty < z < b$ while $\{\varphi_t\}$, the states of the tip, are well described in the range $a < z < \infty$. This is certainly true if a is far from b and in practice remains a good

approximation for many tunneling systems.[55,56] In this case the matrix elements can be expressed (switching to atomic units):

$$T_{st} = -\frac{1}{2} i \int d\mathbf{S} \cdot \left[\varphi_t^* \nabla \varphi_s - \varphi_s \nabla \varphi_t^* \right] \tag{5.2}$$

where the integral is over the surface lying between the two electrodes.

5.1.2.2. One Electron Tunneling Matrix Elements

If we have a description of two states obtained, for example, by a Hartree-Fock computation, we can evaluate the matrix elements in (5.2) for any pair of molecular orbitals. To see this, first we express the integrand in terms of the basis functions:

$$\begin{aligned} \varphi_t^* \nabla \varphi_s - \varphi_s \nabla \varphi_t^* &= \varphi_t \nabla \varphi_s - \varphi_s \nabla \varphi_t \\ &= \left(\sum_{i=1}^{F_t} c_i^{(t)} \chi_i \right) \nabla \left(\sum_{j=1}^{F_s} c_j^{(s)} \chi_j \right) - \left(\sum_{j=1}^{F_s} c_j^{(s)} \chi_j \right) \nabla \left(\sum_{i=1}^{F_t} c_i^{(t)} \chi_i \right) \\ &= \left(\sum_{i=1}^{N_t} a_i G_i \right) \nabla \left(\sum_{j=1}^{N_s} b_j G_j \right) - \left(\sum_{j=1}^{N_s} b_j G_j \right) \nabla \left(\sum_{i=1}^{N_t} a_i G_i \right) \end{aligned} \tag{5.3}$$

where in the first step we have dropped the complex conjugates because the molecular orbitals are real functions. Subsequently, we expand each orbital in contracted basis functions and finally in terms of normalized primitive Gaussian functions.[57] Ignoring constants for the moment, we see that every term in the expression is of the form:

$$g(x,y,z;A,\alpha_A,l_A,m_A,n_A) \frac{\partial}{\partial x} g(x,y,z;B,\alpha_B,l_B,m_B,n_B),$$

or $\frac{\partial}{\partial y}$, or $\frac{\partial}{\partial z}$.

---

[55]N. D. Lang, *Phys. Rev. B* **34** 5947 (1986).

[56]N. D. Lang, *Phys. Rev. Lett.* **55** 230 (1985).

[57]The notation for molecular orbital expansions and gaussian basis functions is thoroughly reviewed in the Appendices.

Using Eq. (0.12) we perform the derivatives. The result is that every new term is simply a product of Gaussians. If we combine the products using Eq. (0.13) we find that the integrand of Eq. (5.3) becomes a single sum over Gaussian functions (having kept track of all the appropriate constant coefficients):

$$\varphi_t^* \nabla \varphi_s - \varphi_s \nabla \varphi_t^* = \sum_i d_i g_i . \tag{5.4}$$

We can now complete the surface integral by substituting Eq. (5.4) into Eq. (5.2) and choosing a surface $z = S_z$:

$$\begin{aligned} T_{st} &= -\frac{1}{2} i \int d\mathbf{S} \cdot \left[\sum_k d_k g_k \right] \\ &= -\frac{1}{2} i \sum_k d_k \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \left[g_k(x, y, z = S_z; P_k, \alpha_k, l_k, m_k, n_k) \right] \\ &= -\frac{1}{2} i \sum_k d_k \left[(S_z - P_z)^n e^{-\alpha (S_z - P_z)^2} \int_{-\infty}^{\infty} (x - P_x)^l e^{-\alpha (x - P_x)^2} dx \int_{-\infty}^{\infty} (y - P_y)^m e^{-\alpha (y - P_y)^2} dy \right]_k \end{aligned}$$

where the subscript k outside the last square brackets indicates the P components, l, m, n, and α are to be indexed by the current term in the summation. Many of the terms are zero: only terms for which both l and m are even survive. By using Eq. (0.4) we can complete the matrix elements

$$T_{st} = -\frac{1}{2} i \sum_k d_k \begin{cases} \left[(S_z - P_z)^n e^{-\alpha (S_z - P_z)^2} \dfrac{\sqrt{\pi}(l-1)!!}{2^{l/2} \alpha^{(l+1)/2}} \dfrac{\sqrt{\pi}(m-1)!!}{2^{m/2} \alpha^{(m+1)/2}} \right]_k & l, m \text{ even} \\ 0 & \text{otherwise} \end{cases}$$

or, slightly rewritten,

$$T_{st} = -\frac{1}{2} i \sum_{k \ni l_k, m_k \text{ even}} d_k \left[(S_z - P_z)^n e^{-\alpha (S_z - P_z)^2} \frac{\pi (l-1)!!(m-1)!!}{2^{(l+m)/2} \alpha^{(l+m+2)/2}} \right]_k . \tag{5.5}$$

The matrix element evaluation is $O(N^2)$ where N is the number of primitive functions per molecular orbital. More specifically, we can over-bound the number of

terms in this sum if all the original orbital basis functions all have $l, m > 0$. The gradient has three terms and each generates two new Gaussians. Therefore, each of the two gradients generates at most $6N_sN_t$ terms for a total upper bound of $12N_sN_t$ terms. In actuality, many terms are zero because they 1) contain odd functions, or 2) are *s*-functions for which the gradient generates only one new term instead of two, or 3) we have chosen an integration surface that eliminates the *x* and *y* components; therefore, the prefactor is less than 12. To generate an STM image, we would like to calculate the matrix elements at each point in a hypothetical scan. Assuming a square grid of N_g points on an edge, the total upper bound is $12N_g^2N_sN_t$ evaluations of terms like those in Eq. (5.5).

5.1.3. Results

We present here the first results for the calculation of Bardeen tunneling matrix elements from *ab initio* wave functions. The system imaged was the *n*-butyl benzene on graphite model interacting with a Ni_3 tip needle. The Ni_3 consists of three Ni atoms spaced 2.487 Å apart. The basis set used was Ni DZP2 + DIFs plus one more *p*-function scaled out with exponent 0.014053 to provide a well-behaved exponential decay. The contributing states from the sample were either those from the 2 eV window or the 4 eV window as described previously. The singly occupied orbital of the tip was used for the tip state. The Ni_3 was oriented perpendicular to the graphite plane and scanned laterally in the planes indicated below with the closest atom as the reported tip-sample separation.

Compared to the total density (constant height) simulations shown previously, Figures 53 to 55 show a greater sensitivity to the edge of the cluster. Comparing the 2 eV sample state window to the 4 eV sample state window shows a filling out of the current near the center of the surface. Figure 55 makes an interesting comparison with Figure 53 since the tip now has only *p*-basis functions. There is a general sharpening of the image.

Figure 56 shows an image of the adsorbate. This is similar to the total density plots. A similar plot done at a distance of 1.0 Å from the ring plane looks essentially the same and is not shown.

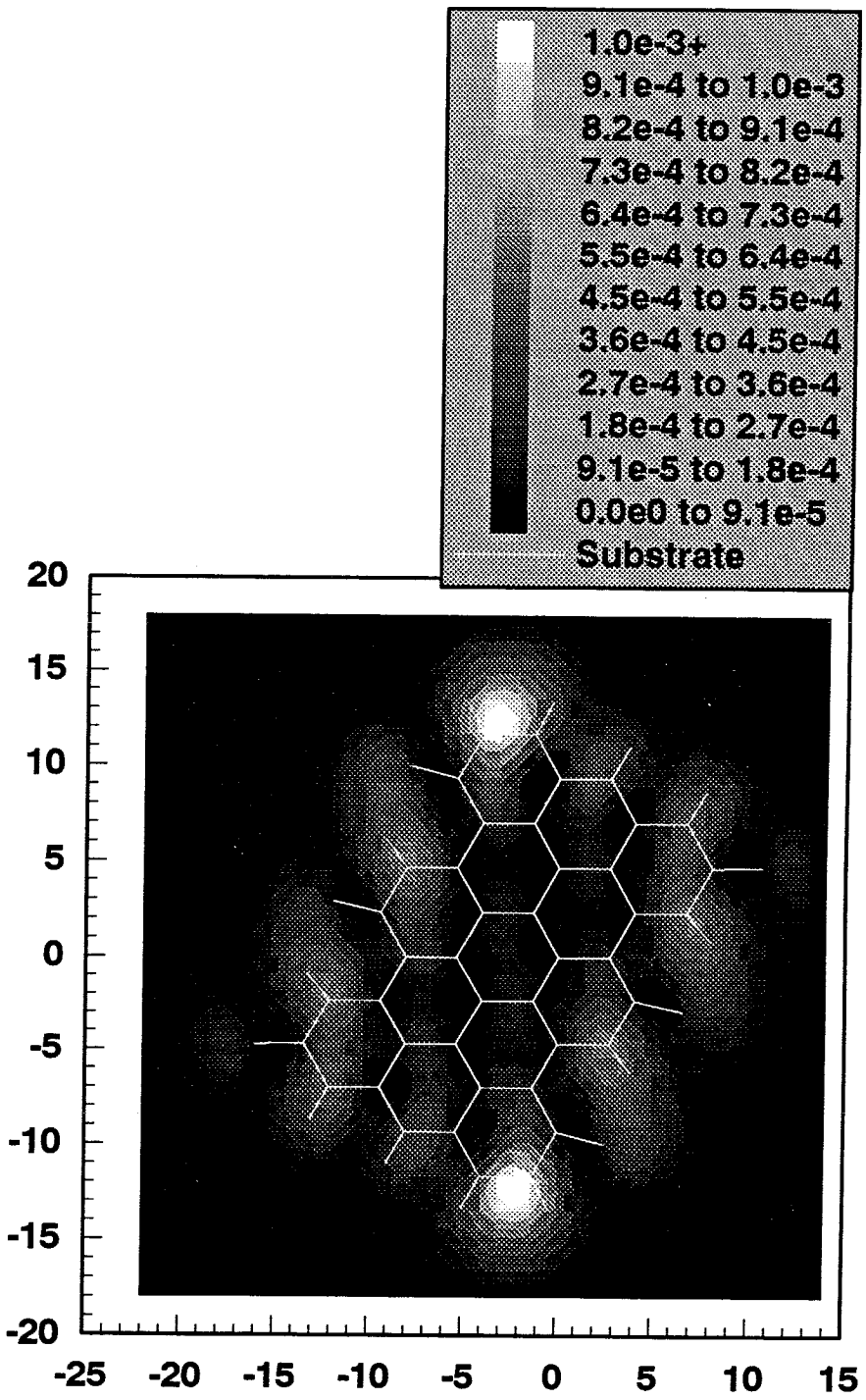


Figure 53:

Bardeen tunneling matrix element (actually, $4T_{st}^2$, atomic units) between HOMO of Ni_3 and top seven occupied states (2 eV window) of the graphite system. Scan plane was 2.5 Å above graphite plane on the side opposite the *n*-butyl benzene adsorbate.

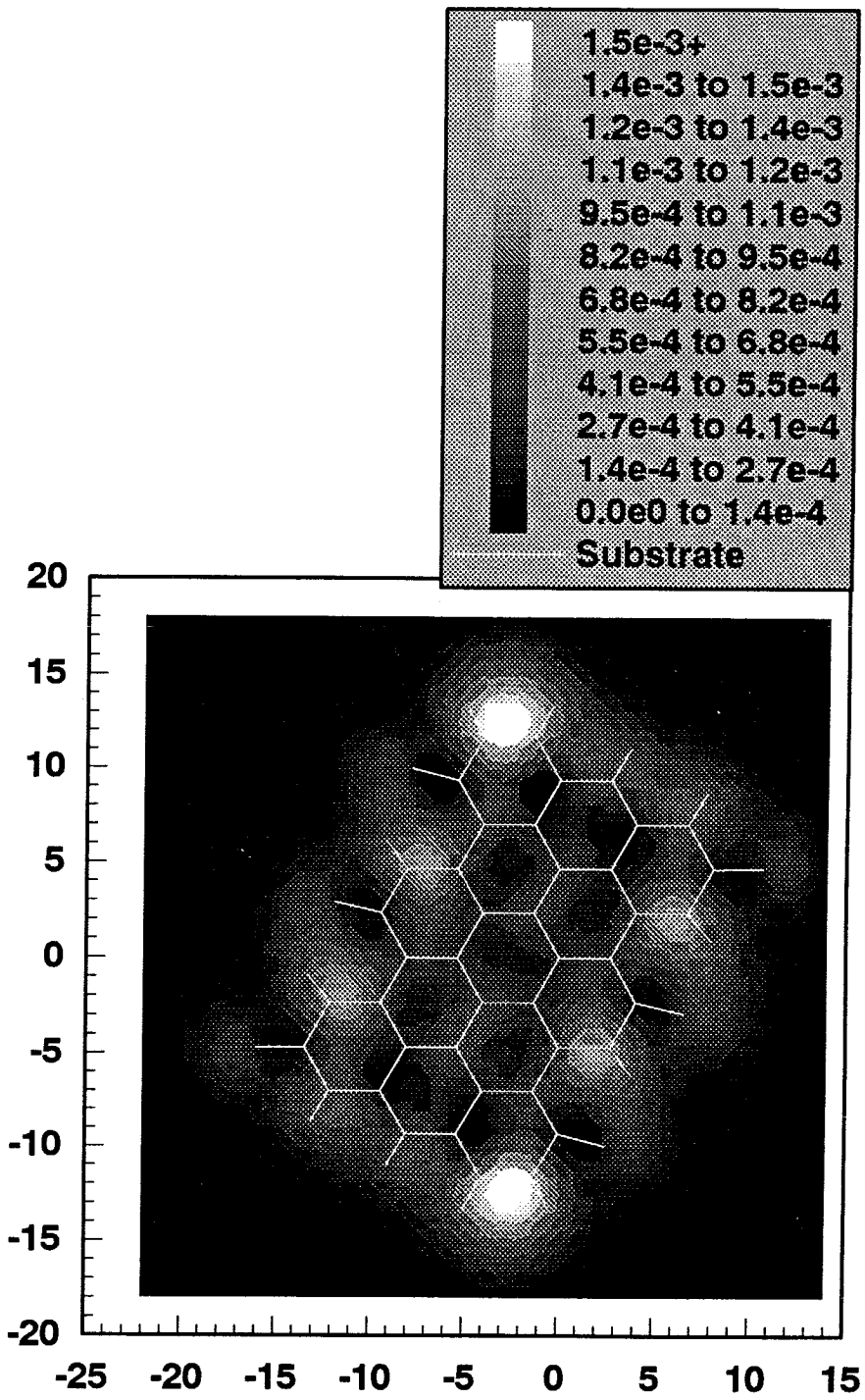


Figure 54:
Similar to Figure 53 but with the top 11 states (4 eV window) of the sample contributing.

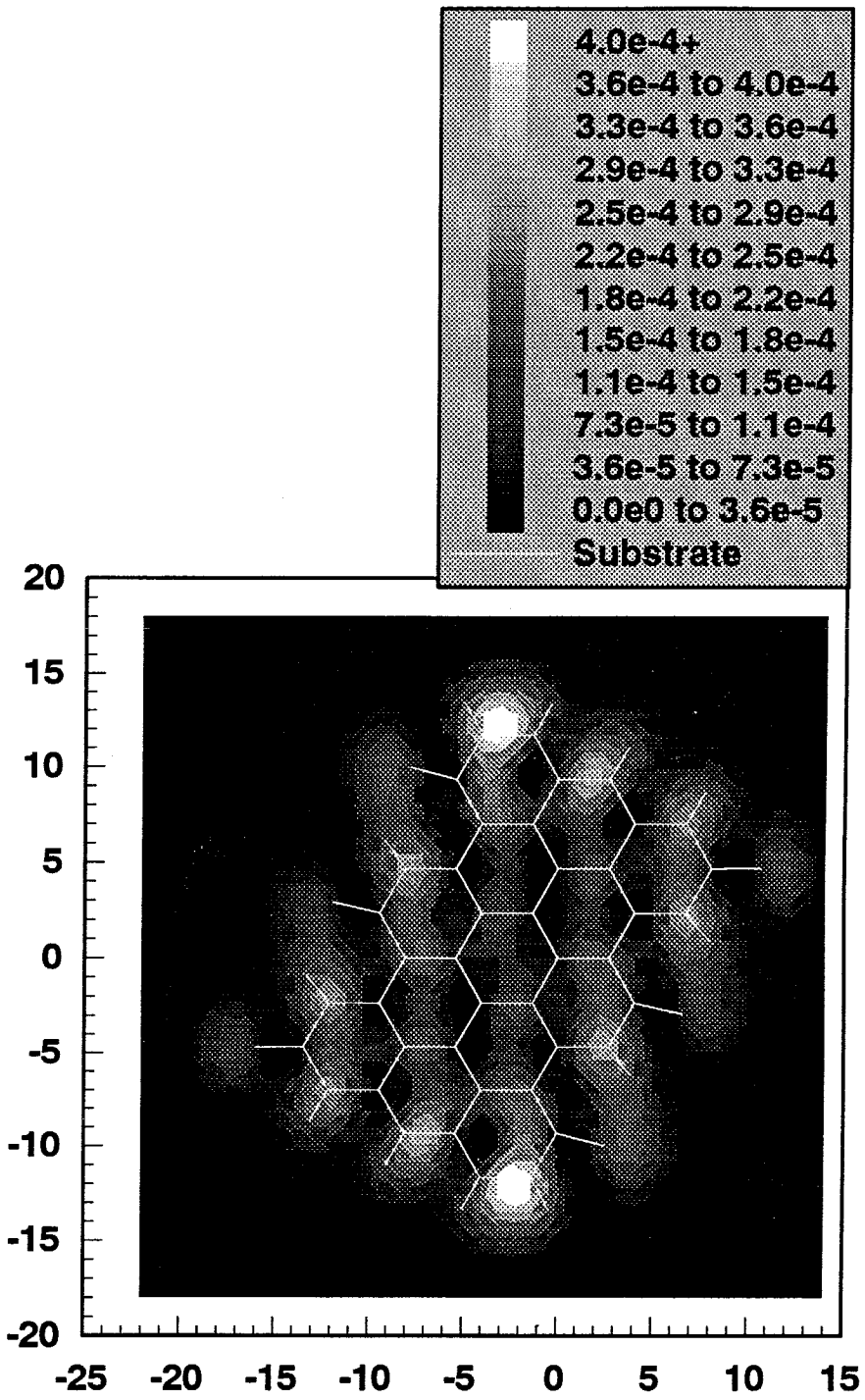


Figure 55:
Similar to Figure 53, however, only *p*-basis functions were present on the Ni tip.

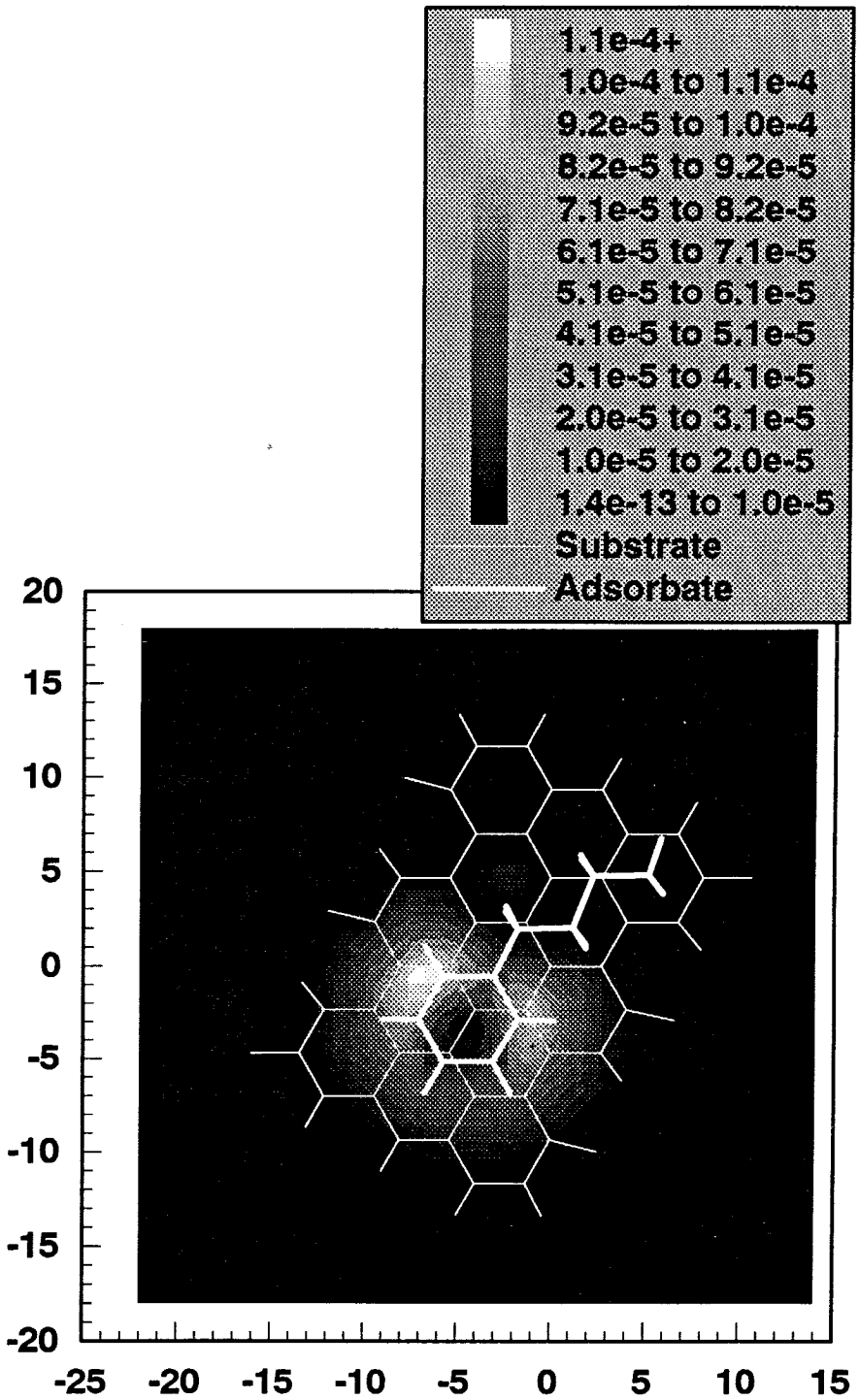


Figure 56:
Similar to Figure 53 except the tip is scanned 3.0 Å above the adsorbate ring plane.

5.1.4. Discussion

A qualitative difference between the total density derived images and the Bardeen tunneling matrix element images is the sensitivity of the Bardeen plots to the edges of the cluster. This is most likely due to the presence of the derivative in the matrix element expression. Near the cluster edges, the wave function z-gradient is necessarily greatest.

Also intriguing is the sharpening seen when only Ni p-basis functions were used. Again, this is due to the derivative in the matrix element expression. Chen[54] has presented a nice discussion of this effect by expanding the tip state in spherical harmonics. Apparently, however, ours is the first application to a chemically specific STM model.

The method we have developed needs additional testing, though results to date are promising. Although we used an extended basis set on the Ni tip, we were unable to equally augment the sample basis set. The degree of sensitivity of the method to the wave function tail should be explored. We note that it is possible to calculate the matrix element between pairs of orbitals in the same wave function. Therefore, the tip and sample system could be converged together as a neutral system and matrix elements between tip and sample computed. It would be interesting to determine the differences between this integrated approach and the approximation of independent tip and sample wave functions.

Of course, if the method appears to be of general use in STM systems, it might also be applied to through-space electron transfer in protein systems. In general, any system where the electron transfer states are well separated should be amenable to this analysis and computation.

5.2. Approximating Bulk Systems With Cluster Calculations

5.2.1. Introduction

We would like to be able to use cluster calculations which provide a high quality description of relatively small systems (<100 atoms) to make predictions about much larger systems. In particular, for the STM system, we would like to calculate an estimate of the change in the number of states per unit energy near the Fermi level caused by the presence of an adsorbate. Because we specifically wish to deal with non-periodic structures (adsorbates, defects), approaches to bulk systems that rely on periodicity are ruled out.

Presented here is our approach to the problem along with preliminary results. The ability to test these ideas relatively rapidly was made possible by the growing amount of software tools we created for quickly and easily manipulating wave functions in Gaussian basis sets.

5.2.2. Pairwise Fock and Overlap Matrices

5.2.2.1. Basis of the Method

First, we briefly review the concepts underlying SCF (self-consistent field) procedures for molecular quantum mechanics (MQM) that are important to our goal of extracting pairwise electronic information and applying it to other systems. Exact details are found in many references.[6] For simplicity, we discuss the Hartree-Fock equations, however, extensions to related and improved methods (e.g., GVB) should not require much work.

Hartree-Fock programs solve the equation

$$\mathbf{FC} = \mathbf{SC}\varepsilon \tag{5.6}$$

where **F** is the Fock matrix in the basis set of atomic normalized, contracted Gaussian basis functions, **S** is the basis function overlap matrix, **C** is the matrix of molecular orbital coefficients, and ε is the diagonal matrix of orbital energies.

$$S_{\mu\upsilon} = \langle \chi_\mu | \chi_\upsilon \rangle$$

$$\mathbf{C} = \begin{array}{c} \\ basis \end{array} \overbrace{\begin{array}{c} 1 \quad \cdots \qquad\qquad N_{MO} \\ \left\{\begin{array}{c} 1 \\ \vdots \\ \\ N_{basis} \end{array}\right. \left[\begin{array}{ccc} c_{1,1} & \cdots & c_{1,N_{MO}} \\ \vdots & c_{\mu,a} & \\ & \ddots & \\ c_{N_{basis},1} & & c_{N_{basis},N_{MO}} \end{array}\right] \end{array}}^{MO}$$

For simplicity, we will discuss the case where the number of molecule orbitals is equal to the number of basis functions. (In practice, the *s*-combinations of higher angular momentum functions, like $d_{x^2+y^2+z^2}$, are removed from the MO space.)

The elements of the Fock matrix are

$$F_{\mu\upsilon} = \langle \chi_\mu | f(1) | \chi_\upsilon \rangle$$

where *f* is the Fock operator. The Fock operator depends on the coordinates of a single electron, arbitrarily labeled "1." It contains the kinetic energy, the potential energy due to the nuclear attraction, and the potential energy due to repulsion with all the other electrons in the system. To determine the potential energy with respect to other electrons a particular distribution of electron density is assumed. Thus, the Fock operator has a functional dependence on a presumed distribution of electrons. Having defined the Fock operator and determined the Fock matrix elements, Eq. (5.6) may be solved for a set of eigenvectors (molecular orbitals) and eigenvalues (orbital energies). A set of MOs can be generated for any proper Fock operator (distribution of charge).

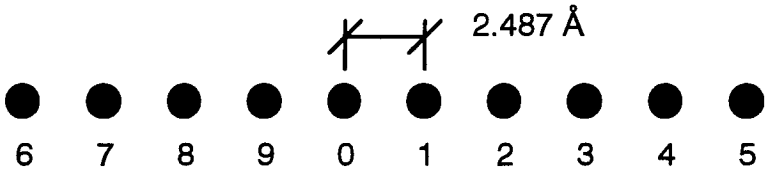
The SCF (self-consistent field) procedure consists of guessing an electronic distribution, creating the Fock operator, diagonalizing the Fock matrix, and then determining a new Fock operator (and Fock matrix) from the set of MOs just found. This procedure is iterated until the distribution of charge no longer changes from one iteration to the next. In practice this stationary point in the energy as a function of the MO coefficients is usually the global minimum (though sometimes local minima, i.e., excited states, can be found). Therefore, at the end of the calculation we have a set of MOs describing the ground state distribution of charge in the system.

If we are given a set of MOs and orbital energies with the basis set for a converged calculation, we can regenerate the Fock matrix if we recompute the basis set overlap matrix and find the inverse of the coefficient matrix:

$$\mathbf{F} = \mathbf{SC\varepsilon C}^{-1}.$$

Since this information is available from almost any Hartree-Fock program, we can obtain wave functions from any of these sources and use our tools for extracting the Fock matrix information.

Next, we identify atomic pairwise interactions that we believe to be chemically well described. As a concrete example, consider the case of a cluster calculation for a linear chain of 10 Ni atoms using the Ni DZP1 basis set:



The Ni DZP1 basis set consists of two *s*-functions and one *p*-function, and the Fock matrix has the structure

$$
\mathbf{F} = \begin{array}{c|cccc}
 & \mathrm{Ni}_6 & \mathrm{Ni}_0 & \mathrm{Ni}_1 & \mathrm{Ni}_5 \\
\mathrm{Ni}_6 & & & & \\
 & \ddots & & & \\
\mathrm{Ni}_0 & & [\mathrm{Ni}_0,\mathrm{Ni}_0] & [\mathrm{Ni}_0,\mathrm{Ni}_1] & \\
\mathrm{Ni}_1 & & [\mathrm{Ni}_1,\mathrm{Ni}_0] & [\mathrm{Ni}_1,\mathrm{Ni}_1] & \\
 & & & & \ddots \\
\mathrm{Ni}_5 & & & &
\end{array}
$$

where, for example, $[\mathrm{Ni}_0,\mathrm{Ni}_1]$ represents

$$
\begin{array}{c|ccccc}
 & s_1^1 & s_2^1 & p_x^1 & p_y^1 & p_z^1 \\
s_1^0 & \langle \chi_{s_1^0} | f | \chi_{s_1^1} \rangle & & \cdots & & \langle \chi_{s_1^0} | f | \chi_{p_z^1} \rangle \\
s_2^0 & & \ddots & & & \\
p_x^0 & \vdots & & & & \vdots \\
p_y^0 & & & & & \\
p_z^0 & \langle \chi_{p_z^0} | f | \chi_{s_1^1} \rangle & & \cdots & & \langle \chi_{p_z^0} | f | \chi_{p_z^1} \rangle
\end{array}
$$

The overlap matrix has the same structure.

The submatrices $[\mathrm{Ni}_0,\mathrm{Ni}_0]$ and $[\mathrm{Ni}_0,\mathrm{Ni}_1]$ contain the Fock matrix elements between basis functions on the same center and nearest neighbors, respectively. This particular way of blocking the Fock matrix is simply a convenient way to group matrix elements that we expect to be dominant. Therefore, we expect the most important contributions to the energy from the same-center terms, $[\mathrm{Ni}_0,\mathrm{Ni}_0]$, followed by nearest neighbors, $[\mathrm{Ni}_0,\mathrm{Ni}_1]$, then second nearest neighbors, $[\mathrm{Ni}_0,\mathrm{Ni}_2]$, etc. That this is the case is intuitively apparent when we recall that the atomic basis functions are designed to be good descriptions of the atomic eigenfunctions. Therefore, *f(1)* acting on such a function should have a large projection on the same function. Similarly, nearest neighbor pairwise combinations of atomic functions describe bonds, and so we expect these to be the next dominant contributors.

If the chain were infinite, the Fock matrix elements in $[\mathrm{Ni}_0,\mathrm{Ni}_1]$ would be the same as those for any other nearest neighbor interaction. The kinetic energy part of the Fock

operator is the same independent of chain length. Therefore, the difference between $[Ni_0,Ni_1]$ for the infinite and finite chains comes from the potential terms. The first potential term is the nuclear attraction, the other terms (coulomb and exchange) arise from electron repulsion with the averaged fields of all the other electrons in the system. Thus, to the extent that $[Ni_0,Ni_1]$ is a good approximation to the infinite chain, we can use these matrix elements to construct a super Fock matrix for a very long chain.

As the chain gets larger, the nuclear attraction and electron repulsion tend to cancel for distances far from a chosen atom because the system is neutral. Thus, we expect the pairwise Fock matrix elements to converge on values in the infinite system even for finite size clusters. This can be seen also by examining the form of the coulomb and exchange operator portions of the Fock operator:

$$\hat{J}_a(1)\chi_\mu(1) = \left[\int d\bar{q}_2 \varphi_a^*(2)\frac{1}{r_{12}}\varphi_a(2)\right]\chi_\mu(1)$$

$$\hat{K}_a(1)\chi_\mu(1) = \left[\int d\bar{q}_2 \varphi_a^*(2)\frac{1}{r_{12}}\chi_\mu(2)\right]\varphi_a(1)$$

Here the $\{\varphi_a\}$ are the set of molecular orbitals optimized in the cluster calculation. Recall that the MOs are in general delocalized functions, while the basis functions are always localized on a particular center. At worst, the energy contribution from the coulomb operator falls off as $1/r$ where r is the distance from the center of basis function μ, and this determines how far we must go (in terms of pairwise Fock operators) to capture its effect within a given error. In actuality, the coulomb field of the MO is screened by the nuclear centers (or vise-versa, but the affect is the same) and falls off more quickly.

The exchange term arises because the total wave function is an antisymmetrized product of molecular orbitals. Matrix elements between permutations of electron

coordinates in different MOs lead to terms of this form. Again, because of the finite extent of the basis function, this term has a finite extent.

5.2.2.2. Systematic Approach

We have developed very general tools to begin testing these ideas. The following is an outline of our approach :

1. Define the large system to be studied.
2. Identify chemically significant and distinct regions of the system.
3. Perform one or more cluster calculations to model the important interactions in the system.
4. Identify the atoms and pairwise interactions thought to be well described in each cluster model. The programs will then extract the relevant pairwise Fock and overlap matrices.
5. Given the geometry of the large system, the programs will next construct super Fock and overlap matrices using cluster matrix elements where appropriate.
6. Next, the super Fock matrix is diagonalized resulting in a set of MOs and orbital energies for the large system.

For example, to model a chemisorbed species, the first few layers of the substrate along with an adsorbate would comprise one cluster calculation. Another cluster calculation would be used to model the interior of the substrate. Fock matrix elements involving all the atoms of the adsorbate along with the nearest surface atoms will be used in the large system. Selected atom pairs from the center of the bulk cluster model will be used to model bulk atoms in the large system. A specific example of this is described below.

5.2.3. Fock Matrix Elements for a Linear Chain of Ni Atoms

Following are the Fock matrix elements for a series of *ab initio* Ni chains (atoms along the *z*-axis). Each case was a Hartree-Fock closed shell system using the Ni DZP1 basis set. In each case the atoms labeled Ni_0 and Ni_1 are the central two atoms of the even-length chains. The self terms, $[Ni_0,Ni_0]$, are actually an element by element average of $[Ni_0,Ni_0]$ and $[Ni_1,Ni_1]$. This is because the even numbered clusters present an asymmetric field to each of the central two atoms individually. Averaging symmetrizes the field as it would be in the infinite (large) chain case. The net effect leaves the non-zero matrix elements shown below exactly as they were in either $[Ni_0,Ni_0]$ or $[Ni_1,Ni_1]$ but zeroes off diagonal matrix elements between *s*-functions and *p*-functions since these elements are just the opposite sign in the two non-symmetric matrices.

Ni self-terms:

Ni_6 $[Ni_0,Ni_0]$	s_1	s_2	*px*	*py*	*pz*
s_1	0.3355	-0.4579	0.0000	0.0000	0.0000
s_2	-0.4579	-0.0898	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2287	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2287	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1367

Ni_8 $[Ni_0,Ni_0]$	s_1	s_2	*px*	*py*	*pz*
s_1	0.3327	-0.4605	0.0000	0.0000	0.0000
s_2	-0.4605	-0.0924	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2275	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2275	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1356

Ni_{10} $[Ni_0,Ni_0]$	s_1	s_2	px	py	pz
s_1	0.3322	-0.4605	0.0000	0.0000	0.0000
s_2	-0.4605	-0.0926	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2256	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2256	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1337

Ni_{12} $[Ni_0,Ni_0]$	s_1	s_2	px	py	pz
s_1	0.3308	-0.4618	0.0000	0.0000	0.0000
s_2	-0.4618	-0.0939	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2250	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2250	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1333

Ni_{14} $[Ni_0,Ni_0]$	s_1	s_2	px	py	pz
s_1	0.3306	-0.4619	0.0000	0.0000	0.0000
s_2	-0.4619	-0.0941	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2241	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2241	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1322

Ni_{16} $[Ni_0,Ni_0]$	s_1	s_2	px	py	pz
s_1	0.3297	-0.4627	0.0000	0.0000	0.0000
s_2	-0.4627	-0.0949	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2237	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2237	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1320

The nearest neighbor Fock matrix elements are taken from the central two atoms in the *ab initio* chain. With respect to this pair of atoms the clusters are symmetric; therefore, no symmetrization is needed. Later, when the super Fock matrix is constructed,

we will need [Ni_1,Ni_0] as well as [Ni_0,Ni_1]. These are just mutual transposes, and therefore the super Fock matrix is symmetric.

Ni nearest neighbor interactions:

Ni_6 [Ni_0,Ni_1]	s_1	s_2	px	py	pz
s_1	-0.2409	-0.2538	0.0000	0.0000	0.1744
s_2	-0.2538	-0.1081	0.0000	0.0000	0.0391
px	0.0000	0.0000	-0.0053	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0053	0.0000
pz	-0.1744	-0.0391	0.0000	0.0000	-0.0556

Ni_8 [Ni_0,Ni_1]	s_1	s_2	px	py	pz
s_1	-0.2073	-0.2258	0.0000	0.0000	0.1677
s_2	-0.2258	-0.0862	0.0000	0.0000	0.0285
px	0.0000	0.0000	-0.0066	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0066	0.0000
pz	-0.1677	-0.0285	0.0000	0.0000	-0.0743

Ni_{10} [Ni_0,Ni_1]	s_1	s_2	px	py	pz
s_1	-0.2395	-0.2532	0.0000	0.0000	0.1746
s_2	-0.2532	-0.1083	0.0000	0.0000	0.0388
px	0.0000	0.0000	-0.0060	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0060	0.0000
pz	-0.1746	-0.0388	0.0000	0.0000	-0.0560

Ni_{12} [Ni_0,Ni_1]	s_1	s_2	px	py	pz
s_1	-0.2094	-0.2278	0.0000	0.0000	0.1688
s_2	-0.2278	-0.0882	0.0000	0.0000	0.0296
px	0.0000	0.0000	-0.0070	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0070	0.0000
pz	-0.1688	-0.0296	0.0000	0.0000	-0.0723

Ni_{14} [Ni_0,Ni_1]	s_1	s_2	px	py	pz
s_1	-0.2391	-0.2532	0.0000	0.0000	0.1749
s_2	-0.2532	-0.1087	0.0000	0.0000	0.0389
px	0.0000	0.0000	-0.0064	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0064	0.0000
pz	-0.1749	-0.0389	0.0000	0.0000	-0.0559

Ni_{16} [Ni_0,Ni_1]	s_1	s_2	px	py	pz
s_1	-0.2103	-0.2288	0.0000	0.0000	0.1693
s_2	-0.2288	-0.0893	0.0000	0.0000	0.0302
px	0.0000	0.0000	-0.0073	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0073	0.0000
pz	-0.1693	-0.0302	0.0000	0.0000	-0.0713

There are discernible patterns in the Fock matrix elements as the chain length increases. For example, we note that the self-terms s_1,s_1 matrix elements alternate between a large delta and a small delta as the chain gets longer. On the other hand the nearest neighbor s_1,s_1 matrix elements seem to oscillate within an overall decay envelope. To explain this behavior we examine the symmetry of the occupied MOs for each chain:

Ni_6 MO	Symmetry
+ + +	even
+ -	odd
+ - +	even

Ni_8 MO	Symmetry
+ + + +	even
+ + - -	odd
+ - - +	even
+ - + -	odd

etc.

Each new occupied state adds a node. For increasing chain lengths one new even or odd state is added. Because the ground state is even, there is always either an equal number of even and odd states, or one more even state than odd. Since in large chains the ratio between even and odd states approaches unity, this immediately suggests we should use small chains with an equal number of even and odd states.

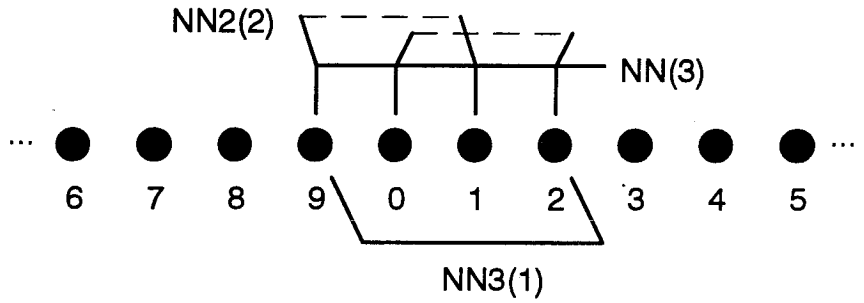
When a new odd state is added to the chain, there is a node at the center of the chain. This reduced electron density deshields the central two nuclei and leads to a greater nuclear attraction component in the field terms of the Fock operator. This is likely the reason we see the largest improvements (decreases) in the *s-s* $[Ni_0,Ni_0]$ matrix elements when a new odd state is added. Consequently, we can make an argument for choosing atoms away from the center for the self-interaction terms, or perhaps for averaging matrix elements from several sites nearer to the center than the ends. We will return to this point shortly.

With respect to the pairwise interactions, we find that having chains with an extra even state improves the $[Ni_0,Ni_1]$ matrix elements substantially. Since we have chosen the pairwise interaction with respect to the center of the chain, it is likely that the dominant effect is an increase in the exchange term in the Fock operator. Since the exchange terms enter with a minus sign, this improves the energy. Again, this argues for the use of chains with an even number of MOs and for averaging the nearest neighbor interactions for a few sites near the center of the chain.

5.2.3.1. Tests for Convergence

In addition to finding the optimal pairwise interactions for each neighbor type, we need to know how many neighbors to keep. The following table shows Ni_{12} reproduced with a Fock matrix reconstructed solely from pairwise interactions extracted from the original Ni_{12} *ab initio* calculation. Matrix elements were created using the guidelines

established previously: Ni_{12} has an equal number of even and odd states; the self interaction was obtained by averaging over the center four atoms; the nearest neighbor interaction was obtained by averaging over the three nearest neighbor interactions available from the center four atoms; similarly, the second nearest neighbor was averaged over the two second nearest neighbor interactions available from the central four atoms; and the single third nearest neighbor interaction between these four atoms was used.



One additional calculation with an NN4 interaction averaged between the elements of $[Ni_9,Ni_3]$ and $[Ni_8,Ni_2]$ was done.

ab initio	NN4	NN3	NN2	NN	Ni
-0.34065	-0.34087	-0.34216	-0.34775	-0.32914	-0.22664
-0.33156	-0.33418	-0.33448	-0.33633	-0.32255	-0.22664
-0.31553	-0.32040	-0.31954	-0.31709	-0.31104	-0.22664
-0.29236	-0.29672	-0.29528	-0.28982	-0.29431	-0.22664
-0.26218	-0.26096	-0.26009	-0.25436	-0.27213	-0.22664
-0.22747	-0.21255	-0.21325	-0.21037	-0.24374	-0.22664

Table 9:
Convergence of state energies as a function of pairwise interactions included.

In going from NN3 interactions to NN4 interactions, the energy change of each state is in the millihartree range (27 meV). The differences from the full calculation are in the vicinity of 5 mhartree with the exception of the HOMO. In the full calculation for the finite chain, the terminal sites which are high energy can relax. In the new Fock matrix

which we have reconstructed from interactions near the center of the chain, this relaxation is not accounted for.

5.2.3.2. Ni Chain

A calculation was performed for a 100 atom chain at the NN3 level using the methods described above for obtaining the Fock matrix elements. The resulting LUMO energy is -0.344419 hartree and the HOMO is -0.179424 hartree. Figure 57 shows the band of Ni *s*-states created. For comparison, the self and nearest neighbor Fock matrix elements taken from the Ni_{12} cluster are shown here:

Ni_{12} $[Ni_0,Ni_0]$	s_1	s_2	*px*	*py*	*pz*
s_1	0.3313	-0.4613	0.0000	0.0000	0.0000
s_2	-0.4613	-0.0935	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2247	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2247	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1328

Ni_{12} $[Ni_0,Ni_1]$	s_1	s_2	*px*	*py*	*pz*
s_1	-0.2294	-0.2448	0.0000	0.0000	0.1728
s_2	-0.2448	-0.1018	0.0000	0.0000	0.0358
px	0.0000	0.0000	-0.0065	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0065	0.0000
pz	-0.1728	-0.0358	0.0000	0.0000	-0.0613

The same calculation was performed using Fock matrix elements extracted from a Ni_{16} chain. As can be seen below, the matrix elements have converged to within a millihartree comparing to the Ni_{12} case. Using these matrix elements, the Ni_{100} LUMO energy is -0.345569 hartree and the HOMO is -0.180335 hartree, again different in the

millihartree range from the previous case. A graph of the band of states is indistinguishable from that in Figure 57.

Ni_{16} $[Ni_0,Ni_0]$	s_1	s_2	px	py	pz
s_1	0.3301	-0.4623	0.0000	0.0000	0.0000
s_2	-0.4623	-0.0946	0.0000	0.0000	0.0000
px	0.0000	0.0000	0.2235	0.0000	0.0000
py	0.0000	0.0000	0.0000	0.2235	0.0000
pz	0.0000	0.0000	0.0000	0.0000	0.1316

Ni_{16} $[Ni_0,Ni_1]$	s_1	s_2	px	py	pz
s_1	-0.2295	-0.2451	0.0000	0.0000	0.1731
s_2	-0.2451	-0.1023	0.0000	0.0000	0.0360
px	0.0000	0.0000	-0.0068	0.0000	0.0000
py	0.0000	0.0000	0.0000	-0.0068	0.0000
pz	-0.1731	-0.0360	0.0000	0.0000	-0.0609

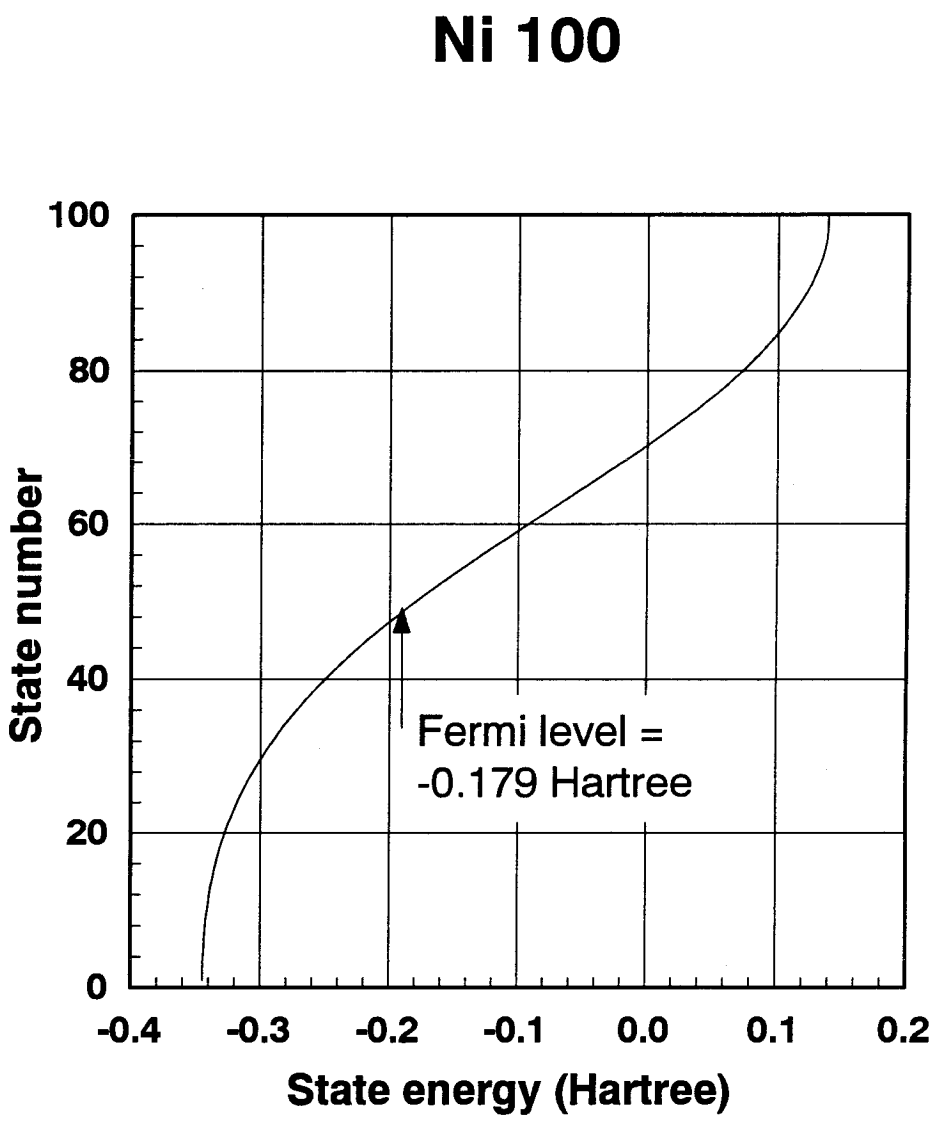


Figure 57:
Linear Ni_{100} calculated from a full Hartree-Fock calculation on Ni_{12}.
First, second, and third nearest neighbors from the Hartree-Fock solution were used in constructing the Ni_{100} Fock matrix.

5.2.3.3. Xe Above a Ni Chain

We have tested a case where an adsorbed Xe atom is present above a chain of 150 Ni atoms. The Fock matrix elements for the Xe-Ni interaction were taken from a Ni_3Xe cluster where the Ni atoms are 2.487 Å apart and the Xe is 3.0 Å above the central Ni atom. The Ni DZP1 and Xe DZ basis sets were used. Fock matrix elements for the single Xe-Ni nearest neighbor interaction were extracted from this cluster and installed in a super Fock matrix built for a Ni_{150} chain. The Ni_{150} super Fock matrix was created from an *ab initio* Ni_{16} cluster using up to NN3 interactions. Averaging was performed as described above.

The resulting Fock matrix that included elements for the interaction of a Xe atom with the number 75 Ni of the Ni_{150} chain was diagonalized and the eigenvectors examined for the presence of Xe *s* and p_z basis function character. The Xe density was determined from

$$\left\langle (a\chi_{s_1} + b\chi_{s_2}) \middle| (a\chi_{s_1} + b\chi_{s_2}) \right\rangle = a^2 + b^2 + 2abS_{12}$$

where *a* and *b* are the coefficients on the *s*-basis functions in the Xe DZ basis set and S_{12} is the overlap between these basis functions. The contribution of Xe *s*-density as a function of energy is shown in Figure 58 and clearly shows a two mode pattern arising from interaction with the even and odd states of the chain. Figure 59 more clearly shows the total Xe density as a function of energy by averaging contributions from adjacent energy dates. The Fermi level of the metal is at -0.18 hartree and shows enhancement of the Xe density in this energy region. Figures 60 and 61 show the same analysis for the Xe p_z density.

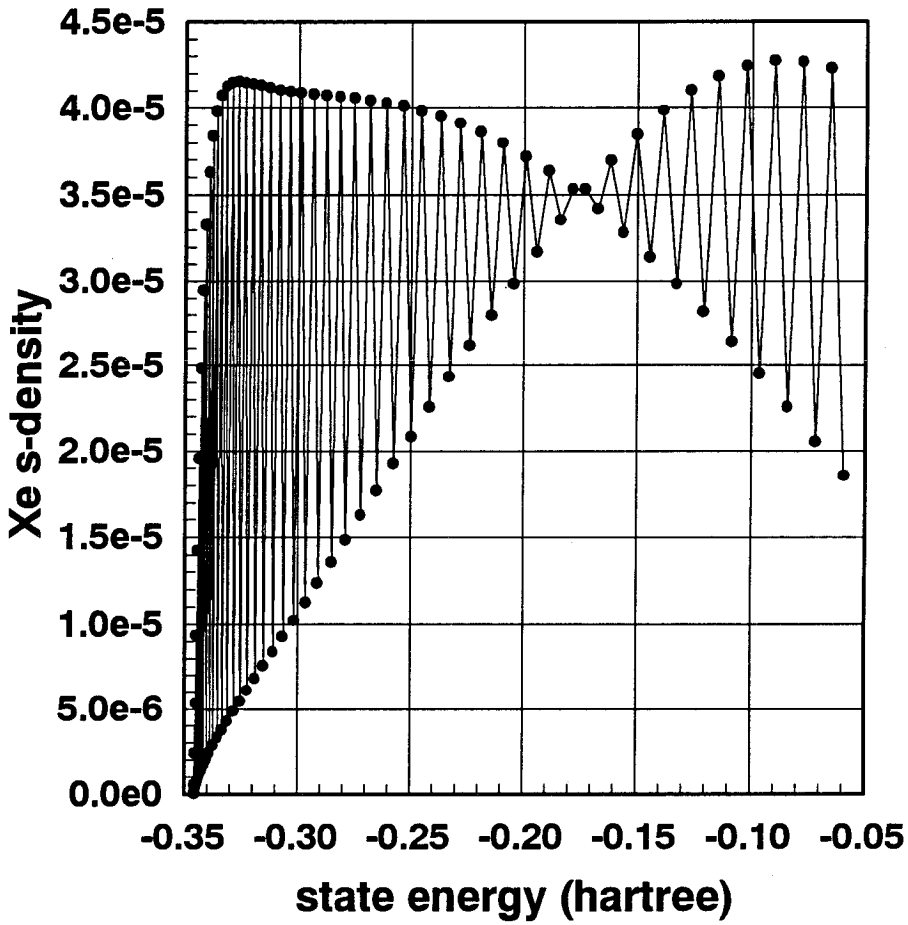


Figure 58:

Xe *s*-density (bohr^{-3}) as a function of eigenvector energy in the $Ni_{150}Xe$ system. Ni Fermi level is at -0.18 hartree.

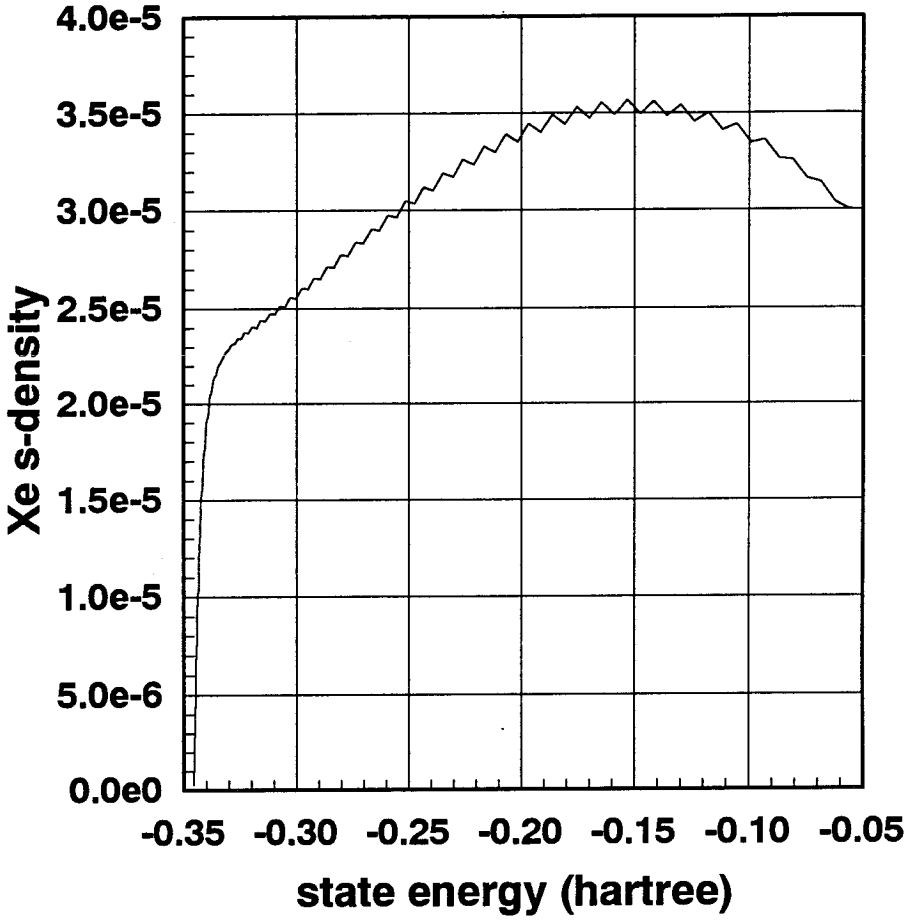


Figure 59:
Similar to Figure 58 but with adjacent states averaged.

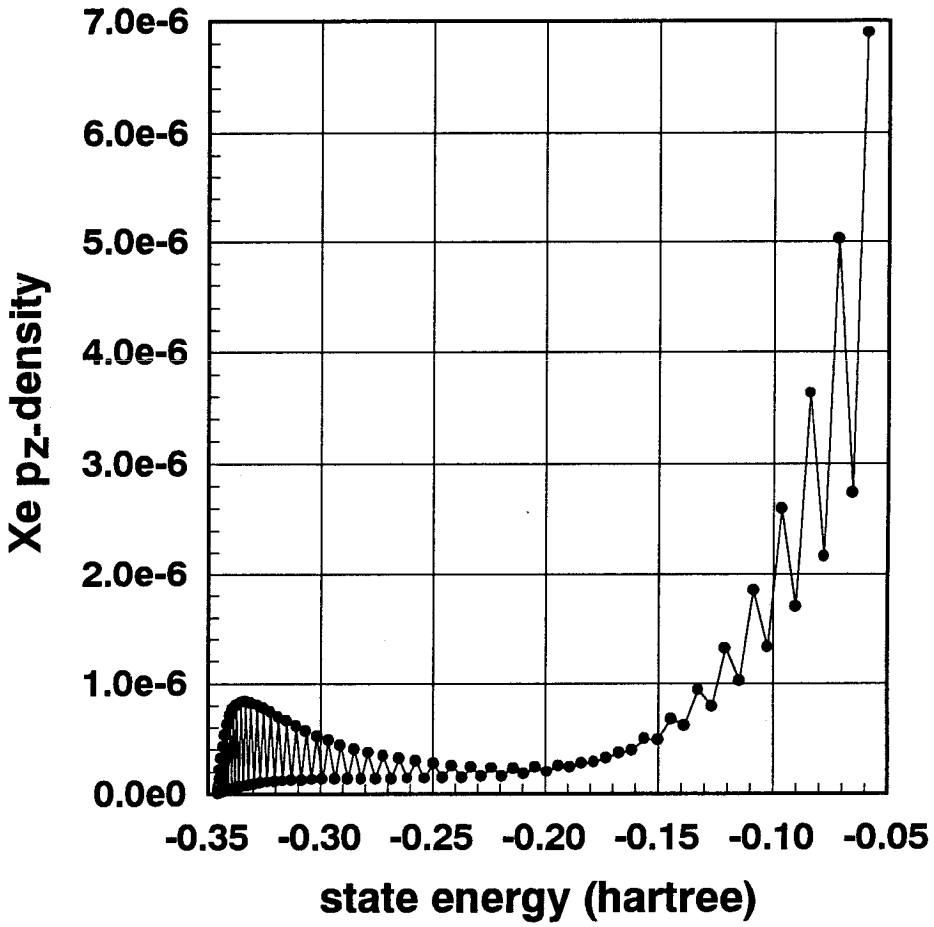


Figure 60:
Similar to Figure 58 but the results are for Xe p_z character.

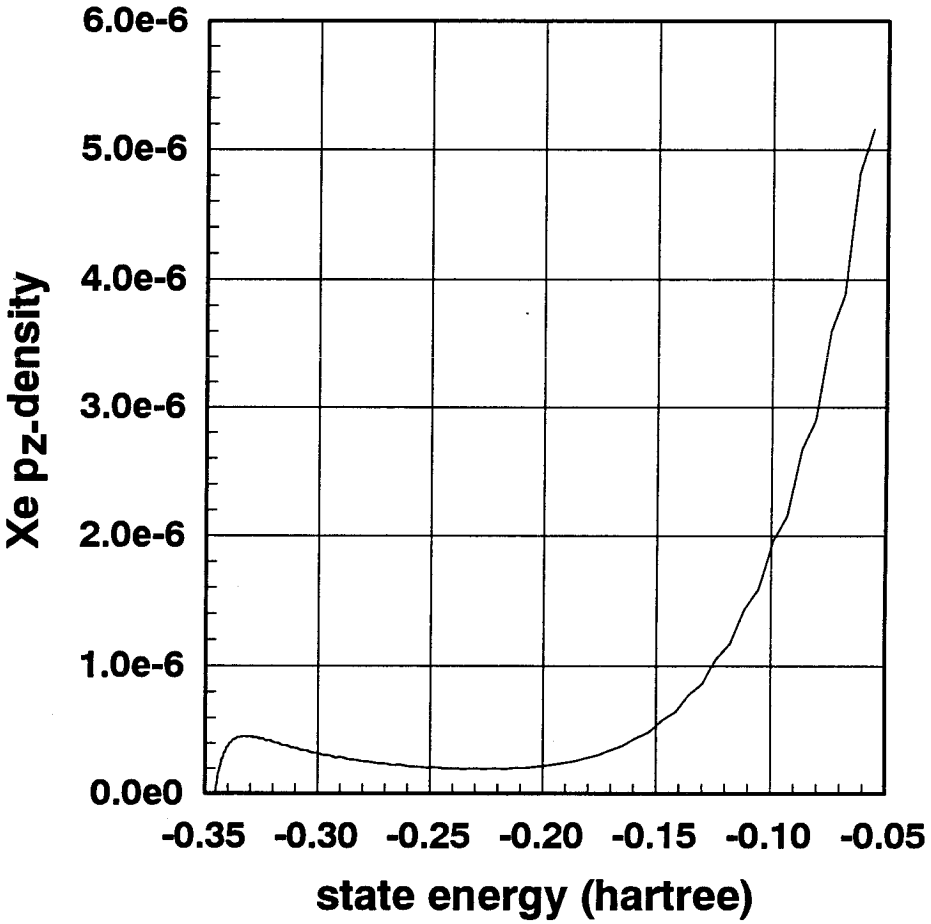


Figure 61:
Similar to Figure 60 but with adjacent states averaged.

5.2.3.4. Discussion

The matrix elements used to generate the super Fock matrix were obtained using a fair amount of detailed knowledge of the linear system. It would be preferable if less analysis were required for obtaining the matrix elements. Based on our experience so far, some general guidelines are suggested:

1. For high symmetry systems, average over several nearly identical interactions. The interactions need not be in the same cluster. If there are a few clusters that give a good chemical description for the environment of a pair of atoms, both can be used and the average taken.
2. A study of the convergence of the Fock matrix elements as a function of the size of the cluster will serve as a guide to the quality of the approximation. An area for further exploration would be to attempt a determination of the asymptotic values for the matrix elements by fitting them as a function of model cluster size.

The results for Xe on Ni_{150} can be improved by adding Xe-Ni second nearest neighbor interactions. The local density of states at the Xe site energies shows an enhancement near the Fermi energy. We believe these results can be extended to yield information needed in the pre-exponential factor of the tunneling expression to complete a quantitative current prediction.

5.2.4. Fock Matrix Elements for an Extended System

In principle, this method can be extended to three-dimensional systems. The increased number of types of chemically unique interactions in the system presents an additional challenge. However, computationally the real bottleneck is the number of

atoms that could be needed in the cluster models to achieve convergence in the pairwise Fock matrix elements. We have developed the tools to manipulate and explore substrate/adsorbate systems. Some initial results are presented.

5.2.4.1. Model

We begin with a good description of the local interactions of an adsorbate with a substrate. Now the task is to embed this description in a much extended description of the substrate in order to study density of states effects more reliably.

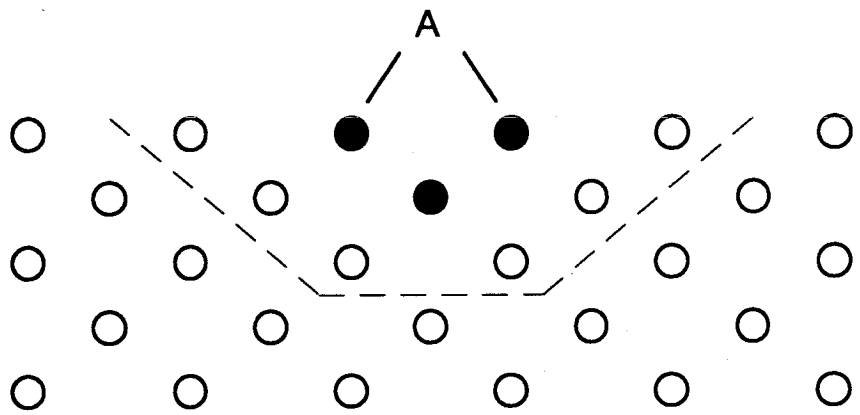


Figure 62:
Schematic of cluster/adsorbate embedded in bulk.

In Figure 62 the atoms above the dashed denote the cluster adsorbate model that has been computed at the Hartree-Fock (or higher) level. In our example this is a Ni_{14}/C_3H_4 cluster. (The geometry is identical to the Ni_{13}/C_3H_4 case but with an additional Ni atom that sits two layers below the Ni at the origin. The cyclopropene thus sits above a rectangular pyramid consisting of 9, 4, and 1 atom layers.) The open circles represent bulk substrate atoms that will be modeled with Fock matrix elements from a pure Ni cluster. Fock matrix elements were extracted from a Ni_{28} cluster. (This cluster is created

by starting with two Ni atoms at the nearest neighbor distance of 2.487 Å and then adding all surrounding atoms such that each original Ni has a full complement of nearest neighbors.) We define three atom types:

Type "A": atoms which are part of the adsorbate.

Type "C": atoms which were part of the cluster/adsorbate system which we maintain and which become part of the bulk. These are the filled circles in Figure 62.

Type "X": atoms which are introduced to extend the substrate.

The type of the atoms involved in a pairwise interaction determines from where the Fock matrix elements are extracted to represent that pair in the super Fock matrix:

Interacting atoms	Origin of Fock matrix elements
A-A	cluster/adsorbate model
A-C	cluster/adsorbate model
A-X	by design there are none of these
C-C	cluster/adsorbate model
C-X	substrate model
X-X	substrate model

5.2.4.2. Test Case

We created a substrate with 96 Ni atoms presenting a Ni (110) surface. The Ni_{14}/C_3H_4 cluster was docked with the substrate and the super Fock matrix created. Ni self interactions and nearest neighbor interactions were obtained from the center of the Ni_{28} cluster. Third, fourth and fifth nearest neighbors were also included using atom pairs

with the corresponding distances in the Ni_{28} cluster. Diagonalization of the super Fock matrix yielded states corresponding well to the adsorbate states of the original cluster. The low energy states of the Ni bulk appeared well described also. However, the Fermi level was too low (ionization potential was at -0.30 hartree, or -8.1 eV). A likely reason for this is that the Fock matrix elements from the cluster had not yet converged to good values for the bulk. At its widest, the Ni_{28} cluster is only four atoms long. In the linear case, we found chains on the order of 12 to 16 were necessary to adequately describe the matrix elements. Another factor is that the Ni_{28} is a high symmetry cluster but there are not enough sites to do averaging as we did for the linear case. A solution to this might be to perform calculations on a number of configurations describing nearest neighbor interactions in the face-centered cubic lattice and average the resulting matrix elements.

5.2.5. A Comment on Nearest Neighbor Approximations

During the development and testing of tools for creating super Fock matrices, we discovered that approximations using standard *ab initio* basis sets and involving only nearest neighbor interactions often lead to negative eigenvalues in the overlap matrix. If one is not careful to correct for this, the energies resulting from solving Eq. (5.6) using the truncated Fock and overlap matrices are inconsistent.

At first, this result was somewhat surprising. Construction of the super Fock and overlap matrices was verified by numerous tests. The most direct affirmation of the occurrence of negative eigenvalues came when we took the analytical overlap matrix for a Ni_3 case (using the Ni DZP1 basis set) and simply zeroed the 1-3 interactions, leaving only nearest neighbor interactions. This matrix compared exactly to the matrix generated by our construction techniques, and does indeed generate a negative eigenvalue. In all our linear chain calculations, negative eigenvalues in the overlap matrix disappear if NN2 (1-3

interactions) or greater are included. We found that more interactions needed to be included for three-dimensional cases.

Negative eigenvalues arise for two reasons: first, the approximate overlap matrix no longer correctly integrates functions expressed in the basis set; second, because the *ab initio* basis set includes *s-p* function interactions (and not simply *s-s* interactions), there are naturally negative overlap matrix elements. For example, whereas *s-s* overlaps are everywhere positive, the overlap of an *s*-function located at (0,0,0) with a p_x-function located at (1,0,0) is negative. We present here the simple procedure used to correct the overlap matrix so that it is consistent with the approximation of nearest neighbor interactions. We note, however, that the presence of negative eigenvalues in the overlap indicates a rather severe approximation is being used. This correction was used in the NN column of Table 9 where consistent results were obtained only with this correction.

Express a function in a basis set, $\{\chi_\upsilon\}$ whose overlap matrix is approximated by **S** and take an inner product:

$$\begin{aligned}
\langle \varphi_a | \varphi_a \rangle &= \left\langle \sum_\mu c_{\mu a}\chi_\mu \middle| \sum_\upsilon c_{\upsilon a}\chi_\upsilon \right\rangle \\
&= \sum_{\mu\upsilon} c_{\mu a}c_{\upsilon a}S_{\mu\upsilon} \\
&= \sum_{\mu\upsilon} c^T_{a\mu}c_{\upsilon a}S_{\mu\upsilon} \\
&= \mathbf{c}_a^{\mathrm{T}}\mathbf{S}\mathbf{c}_a \\
&= \mathbf{c}_a^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\lambda\mathbf{U}\mathbf{c}_a
\end{aligned}$$

where in the last step we have re-expressed the overlap matrix in terms of a diagonal matrix of eigenvalues and a unitary transformation. Choose a direction (coefficient vector) such that

$$\mathbf{Uc}_a = \begin{bmatrix} 1 \\ 0 \\ \vdots \end{bmatrix}.$$

If the eigenvalue $\lambda_{1,1}$ is negative, φ_a has a negative norm. To eliminate the physically unmeaningful portions of the vector space, we zero the negative eigenvalues in λ and reapply the unitary transformation. The result is a reduced dimension space with every available direction having a positive norm.[58]

5.2.6. Conclusions

We have demonstrated that the distances for which pairwise interactions need be included fall off rather rapidly. For our cases no more than 3rd or 4th nearest neighbor interactions need be included in the super Fock matrix. On the other hand, the size of the cluster needed to generate accurate pairwise matrix elements falls off much more slowly. This is fundamentally due to the exponential fall-off in the overlap between neighboring states versus the $1/r$ fall-off in the effect of the field terms in the Fock operator.

Several tricks can be played to generate converged Fock matrix elements. The most important of these for finite cluster sizes appears to be averaging of similar sites to remove biases. Another possibility is to fit the matrix elements as a function of cluster size. Various semi-empirical parametrizations of the coulomb and exchange terms could also be made; the parameters would be varied to match experimental data on the bulk system.

From the standpoint of generating an *ab initio* procedure (here meaning no experimental input) that requires a minimum of finesse (specialized knowledge), our method appears successful for one-dimensional systems. A barrier to success on larger

---

[58]For a further discussion of these issues, see S. A. Cuccaro, Ph.D. thesis, California Institute of Technology (1991). In Cuccaro's thesis basis set error due to numerical inaccuracy of integration is eliminated; this is isomorphic to the current problem.

systems is the ability to generate clusters large enough to contain all the physics. If we make a *crude* estimate based on the linear case needing a chain length of 12 atoms, it is plausible that clusters on the order of 1000 atoms would generate converged Fock matrix elements for bulk systems. Such calculations are on the verge of becoming possible with advances in computing power and new algorithms.[60] Once these calculations are routine, we already have the tools and principles for generating solutions for large, non-periodic systems constructed from independently computed clusters.

We have routinely relied on efficient linear algebra routines[72] for moderate sized matrices. When larger systems are attempted, the nature of our method will lend itself directly to the use of the technology developed for sparse matrix systems on parallel processors.[59]

Of course, we would like to move beyond the Hartree-Fock approximation. By using methods for cluster calculation that incorporate electron correlation effects but still retain the single particle effective Hamiltonian (like GVB), we can directly extend our method to incorporate these results.

---

[59]Bibliographies and commentary on these developments routinely appear on the Internet newsgroup `comp.parallel`; a single example: John R. Gilbert and Hjalmtyr Hafsteinsson, "Parallel Solution of Sparse Linear Systems," SWAT 88 Proceedings (1st Scandinavian Workshop on Algorithm Theory, Halmstad, Sweden, July 5-8 1988), published as Springer-Verlag's "Lecture Notes in Computer Science" #318 (pp. 145-153).

6. A Fresh Approach to Software Design for Computational Chemistry

The problems of developing good scientific research computer programs are very real: the tasks we put to the computer continue to grow in complexity; reliance on older, less well understood code is necessarily increasing (or else we would never get anything new done); add on top of this the advent of parallel processing and the concomitant need to recode old algorithms and develop new ones. But for all this added complexity, the basic mode of operation for the computational chemist has remained the same: edit, debug, compile, link - only the punch card decks are gone, replaced by full-screen editors.

A fresh approach to the problem of developing, maintaining, and using computational chemistry software is in order. We present such an approach, complete with an actual implementation for solving a class of real scientific problems. An attempt is made to strike a balance between the level of detail versus conceptual information presented. The concepts are the most important part, but the detailed examples are necessary for proving the benefits.

The software system we have designed here is novel in many respects, and to our knowledge there is no comparable system in existence. To be sure, pieces exist in various places, but none have been put together with computational chemistry in mind. Part of the reason for this is that some of the tools have only become available in the past few years. Another reason is that computational scientists have been reluctant to invest the time to learn about advances in computer science until the benefits are proven. We focus on case studies to help make our points.

6.1. Motivation

Two factors motivated the extensive software design and implementation project described here. The most direct motivation was the need for tools to move the STM work forward. The second has to do with recognizing structural deficiencies in the way scientific programming tasks are currently approached.

6.1.1. Need for STM Computational Tools

Analysis of the results of our STM calculations is a numerically intensive operation. The three-dimensional nature of the data obtained from STM requires us to explore the electronic properties of our calculated wave functions in three dimensions. In particular, isosurfaces of the wave function density are frequently useful.

We found that most of the tools at hand (programs developed largely in the late '70s to early '80s) were ill-suited to the analyses we required. Worse, the programs were not amenable to modification: it was judged upon reviewing much source code that any such attempt would generate a new program for which we would have serious reservations as to its correctness. The reasons for this are many, but rather than state them here, we believe they will become self-evident in what follows.

With the advent of the pseudospectral Hartree-Fock program[60] we were able to calculate Hartree-Fock wave functions for model systems that are much more realistic in the sense that the model is more chemically intuitive and requires less specialized knowledge of the system (*finesse*). However, these wave functions also require much larger numbers of basis functions putting them absolutely beyond the limits of our current tools for analysis.

---

[60]Murco N. Ringnalda, Jean-Marc Langlois, Burnham H. Greeley, Youngdo Won, Thomas V. Russo, Richard P. Muller, Bryan Marten, William A. Goddard III, and Richard A. Friesner, PS-GVB v0.07, Schroedinger, Inc., 1992.

6.1.2. Structural Problems with Scientific Programming

We could have embarked upon generating the needed software tools using the same approach taken for the existing code: write and debug a monolithic, standalone FORTRAN 77 or C language program. However, the specialized scope of the result compared to the amount effort made this approach unattractive. Instead, we totally rethought the environment for computational chemistry, from both the scientific programmer's standpoint and the scientific user's standpoint. The initial feature set was driven by our STM applications.

Following are some of the fundamental structural problems that currently reduce the efficiency of many scientific programmers. Consequently, these are the goals addressed by our new approaches. Our solutions to these issues are presented in the next section.

6.1.2.1. Leave the Debug-Edit-Compile-Link Cycle

The debug-edit-compile-link (DECL) cycle has been the norm for implementing new algorithms. As the complexity of software grows, this becomes an exceedingly inefficient procedure for software development because an increasing amount of time is spent waiting on the computer to link a large executable for one change, or to load a huge program into the debugger. There exist tools for avoiding these problems (incremental linkers), but they have been slow to move into the scientific research community.

6.1.2.2. Move Scientific Programming to Higher-Levels

Too much time in computational chemistry programming is spent worrying about details. By "details" we mean the tedious and error prone task of getting a program in FORTRAN or C to work correctly.

Progress has been made in many areas to abstract the process of scientific computation above the detailed level of programming. Perhaps leading the way are the symbolic math packages now available.[61,62] These packages allow algebra to be carried out on the computer much as it is on paper, without worrying about correct computer programming (or algebra errors). Most of these packages will also generate computer codes once an algebraic expression is complete. These programs are interactive (no DECL) and the level of interaction is exactly where the user wishes to work (algebra).

Another example of higher level programming is AVS, the Application Visualization System.[63] This system which first appeared in concept around 1987,[64] allows scientific and engineering data to be transformed and finally visualized using a data flow network. The creation of a network is akin to the drawing of a flow chart for the processing of a data set from input to output. We have used this software to produce our STM images from the volumetric wave function property information generated with our programs. Since AVS, other products using this visual data flow paradigm have become commercially successful.

We would like to create an environment for producing efficient computational chemistry programs in a similarly high-level fashion. Our metaphor, however, will be different from both of the examples just described.

---

[61]Maple symbolic algebra software, Waterloo Maple Software, 160 Columbia St. West, Waterloo, Ontario, Canada N2L 3L3; (519) 747-2373; `info@maplesoft.com`

[62]Mathematica symbolic algebra software, Wolfram Research, Inc., 100 Trade Center Dr., Champaign, IL 61820-7237; (217) 398-0700; `info@wri.com`.

[63]AVS is licensed by AVS, Inc.

[64]C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, A. Van Dam, *IEEE Compter Graphics & Applications* **JULY**, p. 30. (1989).

6.1.2.3. Write Robust and Maintainable Software

Much computational chemistry software in its present state is not only difficult to use, maintain, and generalize, but it leads to a lack of understanding of what the software is doing. Unfortunately, almost nothing written today will not need to be changed or enhanced by someone else at a later date. Yet, the unapproachability of much older computational chemistry software has resulted in a body of code some of which has been static for 15 years and inadequate for 8 years. Furthermore, the software often has built-in limitations as to problem size (e.g., number of basis functions) with no obvious way, short of rewriting the entire program, to reliably increase the limits.

6.1.2.4. Improve Software Usability

More often than not, the use of scientific software is effectively restricted to a set of experts who have mastered the intricacies of coaxing old software to solve new problems. In the worst cases use is limited to the developer only.

6.2. Solutions

It is important to keep in mind that we are not referring to "toy" programs here. To solve many of the most interesting problems in computational chemistry requires a complicated problem setup followed by a processes of critical evaluation and recalculation. For a simple Hartree-Fock run the input must contain such diverse information as basis sets, molecular geometry, electronic state information, and perhaps a starting guess. It is the process of developing and using complex applications for which we present a significantly more useful approach.

Finally, much of what we have developed is not limited to computational chemistry. However, all the specific examples pertain to computational chemistry.

6.2.1. Solving the Structural Problems

Here is how the structural problems outlined previously are solved in our approach. Further explanations appear later.

6.2.1.1. Leaving DECL

Without using any commercial products for software development, we have developed an approach that through its design helps reduce time spent in the DECL cycle. The trick is to use *interpreted code*. Interpreters exist in many forms, but what they have in common is that the development cycle is no longer DECL, but rather DE (debug, edit). Interpreters exist for all levels of programming languages, from highly abstract to low level and detailed. The symbolic algebra packages are interpreters, and loosely speaking, so are AVS data flow networks.

What will distinguish our approach is the type of objects that are manipulated by the interpreter - we have developed computational chemistry objects, like Molecules, Grids, and more. Of course, nothing in our approach precludes the use of commercial products to speed the portions of software development that still require traditional programming.

6.2.1.2. High-Level Abstraction

There are two ways we have abstracted the science from the programming details. The first is by the use of an object oriented language, C++. The second is by creating a series of computational chemistry *objects* which are manipulated by an interpreter available to the developer and user. The use of C++ benefits those parts of the programming task that must remain in the DECL cycle, whereas the chemistry object interpreter aids in algorithm design and calculation setup for production runs.

6.2.1.3. Robustness

It is important to recognize the reality that in a research environment documentation is the first part of a software project to perish. Therefore, the source code must be as self-documenting as possible. While nothing can replace a healthy ratio (>1) of comments to source code, object oriented languages lead naturally to much more readable, and therefore more maintainable source code. Readability is enhanced when the object oriented tools for high level abstraction, such as overloading,[65] are taken advantage of. Although good code can be written in C through the use of pointers and data structures, it is hard to write abstract code as easily and effectively as in a language like C++.

To build code that is free from arbitrary size limitations we must choose a language with access to dynamic memory allocation. If a piece of code is general, free from arbitrary limitations, and maintainable, we call it robust.

6.2.1.4. Usability

Using the chemistry object interpreter, we will be able to create code that is free from arcane input formats. We will also discuss tool kits available for creating menu driven applications with a small amount interpreted text.

6.2.2. Overview of Design Features

Having mentioned the features of our design that address current structural difficulties in scientific software development, we proceed to describe the overall design of

---

[65]We can describe overloading with an example: the built in meaning of "*" in most languages is multiply. By default this is valid only for numeric data types. However, multiplication has useful meaning in other contexts. We might like to multiply two Gaussian functions objects named G1 and G2. C++ allows the "*" operator to invoke the method for multiplying Gaussian functions when the source code "G1*G2" appears. The result will be a new Gaussian function object with the exponents correctly combined.

the software. It will become more evident how features we have already described in passing fit into the whole.

6.2.2.1. Programming Language Choice

The majority of the new computational chemistry code we have implemented is written in the object oriented language C++. Nevertheless, portions of the programs are written in ANSI C, and even some FORTRAN 77. When writing a particular section of code, the choice of programming language was dictated by a balance of what would make writing that section of code easiest, most maintainable, and most efficient. Generally, sections of heavy numerical computation will be written in FORTRAN 77 for efficiency reasons - at present it is still true that many computer vendor's FORTRAN compilers generate the fastest code. Most other sections of code involve manipulating data structures in preparation for the numerical part. Handling data structures is most easily accomplished in a high-level object oriented language like C++. We have already mentioned some of the benefits of object oriented programming languages Rather than define object oriented programming here, we will continue to point out the benefits of object oriented languages and approaches in the context of specific examples we have implemented.[66]

We have included a few Language Notes. Their purpose is to demonstrate in context why one language is more appropriate for a particular task than another. These notes are included because this is a question that arises frequently in the computational chemistry field.

---

[66]For an introduction to object oriented concepts in general and the C++ language in specific, see Bjarne Stroustrup, **The C Programming Language, 2nd Ed.**, Addison-Wesley (1991) ISBN 0-201-53992-6

6.2.2.2. Embedded Control Language Interpreter

All the programs we have developed, clients and servers, contain a common embedded interpreter. The embeddable language parser is a public domain package created recently by Dr. John Ousterhout, U. C. Berkeley. The package is a set of libraries which implements a reasonably simple but powerful computer language called the Tool Command Language. We will refer to both the package as a whole and the language it implements as Tcl. A brief introduction to Tcl is needed at this point to clarify its purpose in what follows. For more details, the reader is referred to the excellent set of references that is developing for learning and using Tcl[67].

Tcl defines a simple, extensible computer language that can be used to control the function of a computer program. Inherent in Tcl is a standard set of verbs, control structures (*if-then-else*, *for-loops*, etc.), and support for variables. These features are what make Tcl a computer language. The Tcl package implements an interpreter that reads text written in Tcl. Such a piece of interpreted text is called a Tcl script.

If this were all, there would be no way to make a connection between what the Tcl interpreter is doing (reading a Tcl script and performing the required actions), and the computational chemistry program. Because the verb set of the Tcl language is *extensible*, the programmer can create new commands that will be recognized as legal by the Tcl interpreter. Upon encountering one of these externally defined commands, the Tcl interpreter "calls back" to code written by the application programmer. For example, our fast wave function amplitude generator defines the command `"calculatePrimitiveAmplitudes"` and the callback routine associated with this

---

[67]John K. Ousterhout, Computer Science Division, Dept. of Elec. Eng. & Computer Science, University of California, Berkeley, CA 94720 `ouster@cs.berkeley.edu`, **An Introduction to Tcl and Tk**, Addison-Wesley (1993).

command generates amplitudes for every primitive basis function in an instance of a Molecule object. (The command also takes arguments, one to specify the particular Molecule object, and another to specify a Grid object which holds information about the Cartesian coordinates of the grid points on which to compute the amplitudes. See below.) *Thus, a Tcl-enabled program can interpret standard Tcl scripts which have been augmented with commands specific to the function of the program.*

6.2.2.3. Client-server Model

All the software components developed here are designed to work as either computational "clients" or computational "servers." A computational client requests results and directs the flow of operations to obtain those results. The computational results themselves come from a computational server process. The purpose of a server is to listen for requests to perform specific (computational chemistry) tasks.

6.2.3. New Software from the User's Viewpoint

When the developers are done with a piece of general computational code, it becomes "production code." The user would like to know: how do I interact with "production" scientific software to get answers to my series of problems? One might break down the user's viewpoint into two aspects:

- Program input:

  How is the problem defined for input to the program(s)?

- Program flow:

  How is the program executed? How are programs chained to get a series of results?

Here we contrast the old way of interacting with programs with our new way.

6.2.3.1. Old Way

Program input:

1) With much old software, the user first prepares an input "deck" (file) with a series of "cards" (lines) specifying the problem. Often these decks have an extremely rigid and error prone format. Using these programs is a common source of frustration...

```
[seen on Ohio Supercomputing Center
 computational chemistry mailing list March 30, 1992]
I have been trying to use the program polyrate(version 4.5.1) to
determine rate constants from ab initio data. I have however not
been able to set up an input file that reads in vibrational frequencies
directly. I was wondering if anyone else has used this program and if
they have used the option in the electronic structure input file that
allows frequencies to be read in directly. An example input file of this
type would be much appreciated.
```

2) With some more modern software, calculations can be setup via an interactive menu.

Program flow:

1) The program executable is located and then submitted to the operating system along with the input deck. In some cases this process is encapsulated in a "script" that will accept the name of the input deck file and take care of extra details pertaining to running the program, but the paradigm is the same.
2) With menu driven software (graphical or otherwise) there is a command to start the computation. A few programs have a primitive ability to start a calculation and then let the user out of the menu to recover the answer later. However, in general *there is no conceptual separation between the menu input and the computation*: they are usually the same program.

6.2.3.2. New Way

Program input:

1) The most direct way to setup a problem for computation with one of our servers is to write a Tcl script. Because the work of parsing the input is taken care of by the Tcl interpreter, the format is no longer rigid. In its simplest form a script might consist of a sequence of keyword-value pairs. The keywords are, of course, just Tcl commands defined by the particular server being written to. At the level of script writing, there are two main advantages over creating old-style input decks:

- Input is easy to create and easy to read, and
- Syntax is consistent across servers (and authors of servers).

2) More often than script writing, the user will communicate with the chemistry server via a client program. The client will be responsible for interacting with the user and sending the appropriate Tcl commands to the server. The clients we have developed are graphical menu systems. Creation of these clients is very easy and will be described in the section on the programmer's point of view. We distinguish clearly between the menu input program and the computational server program. There are no special advantages from a user's point of view to using a separate client program or an old-style integrated menu. The advantages will be seen in program flow and software development time.

Program flow:

1) Scripts are executed by a special client which has been written. This client simply reads the script and sends the listed commands to the specified server. At present, the server must be invoked manually by the user (by simply running the server program). For example, if local ionization potentials are requested, the user must be sure this

server is running before running the Tcl script. The goal, however, is to provide a database of services and a way of requesting services by name. The database engine would then be responsible for starting the required servers to complete the calculations requested.[68]

- An important advantage of controlling the flow of calculations with a Tcl script is the availability of the built-in control constructs. For example, if a series of molecules are to be processed in a similar way, the natural thing to do is write a Tcl *for* loop.
- A Tcl script can interact with multiple servers. Since servers share the same syntax, the chaining of results from server to server becomes far easier.

2) At present, our menuing clients control program flow by starting the requested servers and shutting them down when done. To the user, the process is similar to current programs with menu interfaces, with one exception. Because the menu interface client is separate from the server, it is possible to develop clients that disconnect from the server and then reconnect later to collect results. This is useful, of course, if the computation will take some time.

6.2.4. New Software from the Programmer's Viewpoint

The developer of scientific code would like to know: how do I deal with my software work-in-progress to make enhancements, fix problems, etc.? Here we discuss the internal structure of a generic server as we have implemented them.

The following diagram shows the structure of a chemistry server. Blocks of code sharing edges communicate with each other.

---

[68]The UNIX `inetd` provides one way to invoke servers by name. It may be adequate as a first step for handling computational servers also.

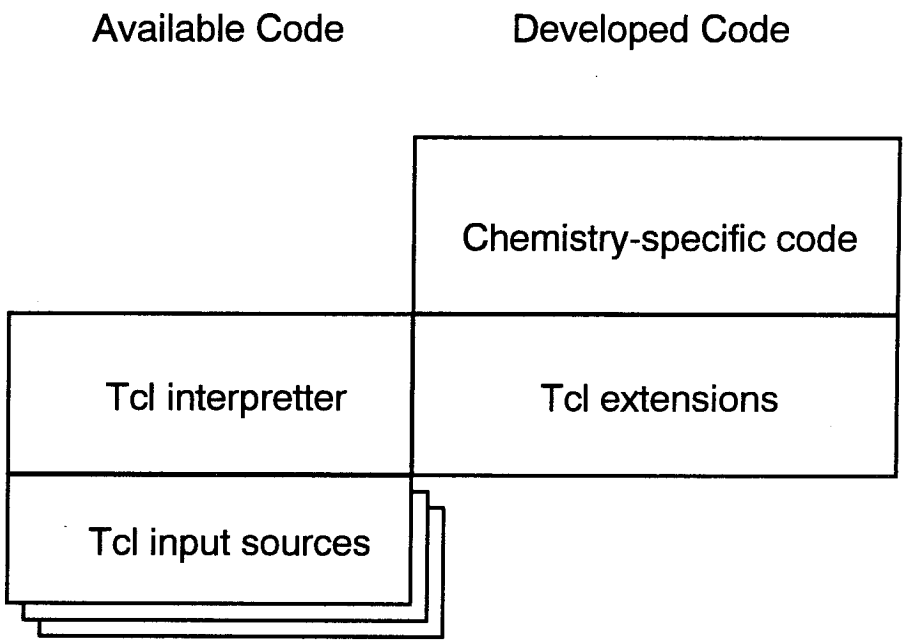


Figure 63:
Block diagram of a computational chemistry server.
Edges connect code blocks that communicate.

6.2.4.1. Program Input

The raw input to the Tcl interpreter can come from many sources. The input could be a routine that reads the "standard input" (e.g., keyboard). In our case all input to servers comes via TCP/IP sockets.[69] We will not describe the details of UNIX sockets here, but in short, sockets provide a means of reading data into a program over a network. With the proper security controls connections can be made between any two computers on a network. For example, the Tcl control stream between a client and a server might bridge two computers on a local network, one displaying results (client) and the other computing results (server). However, the connection can just as easily be between two

[69]Information is available online on any UNIX computer.

computers located across the country (or world), or it can be made between two processes running on the same computer.[70]

There are no special cases the programmer or user need to be aware of when constructing programs or calculations to run across the network. The only practical issues that come into play are communication bandwidth issues. Since remote network connections tend to be slower, the designer of a program or calculation should be aware the communications costs implicit in the particular topology of interconnection that is being considered. Our model primarily addresses client-server control functions, as opposed to data transfer issues. Because control information is usually comparatively small, remote connections are no problem.

Servers have no other apparatus for control input/output. In fact, everything on the left in Figure 63 is code that has already been written; the scientific programmer is responsible for code on the right. Therefore, the work to construct a chemistry server has been made very dense in scientific coding. The details of parsing complicated input are taken care of consistently for all servers by the Tcl interpreter.

6.2.4.2. Program Flow

Each computational server has a specific set of functions it can perform for clients. A client invokes certain servers to be run by the operating system and then makes requests of these servers. Clients may be specific to one type of server, or may interface to multiple servers at once. Furthermore, clients may be as complex as a graphical menu

---

[70]The small package of tools for creating socket connections with easy connections to a Tcl interpretter originated with Kevin B. Kenney at General Electric. The source code is publically available and the locations can be found be reading the newsgroup `comp.lang.tcl`. Kenney's package, which we modified to suit our needs, is an excellent example of the quality tools that can grow up around an idea as general and useful as Ousterhout's Tcl embeddable interpretter. Familiarity with these tools can greatly expedite scientific programming projects.

interface that collects input from the user and issues requests to the server, or *as simple as a text script* written by the user and containing direct commands to be sent to the server. One immediate consequence of this is that the server under development *always has a baseline user interface*, i.e., the simple text script. Once a server is proven to be working and useful, a client that contains more intelligence may be developed to guide users not as familiar as the scientist/developer with the commands implemented by that server. This can even be done easily by someone other than the developer of the server, for example, someone who finds himself using the software very frequently can write his own utility clients.

The separation of function (input in a client, and computational chemistry services in a server) is more than cosmetic. It has a great broadening effect on how the computation can be organized. In particular, *clients need not be on the same computer as the server*, and *clients may organize the activity of more than one server at once.*

The coarse-grained parallelism available for free with our client-server model is well suited to utilizing a network of workstations. However, it is not intended to take the place of network parallel processing tool kits like PVM or Linda.[71] These tools provide built-in mechanisms for network data sharing from the traditional programming function call interface. For example, one may specify that an array of numbers is to be spread out and operated on in each node of a network. In contrast, our client-server model only makes a connection between the input and output of two Tcl interpreters. The details of data and control exchange are up to the user. There is no reason not to create a Tcl-enabled server that uses another parallel package for computational purposes.

---

[71]For information on a variety of public domain and commercial parallel processing packages, access newsgroup `comp.parallel` on the Internet. Access instructions to the "parlib" library of online information on parallel processing is posted to this group frequently.

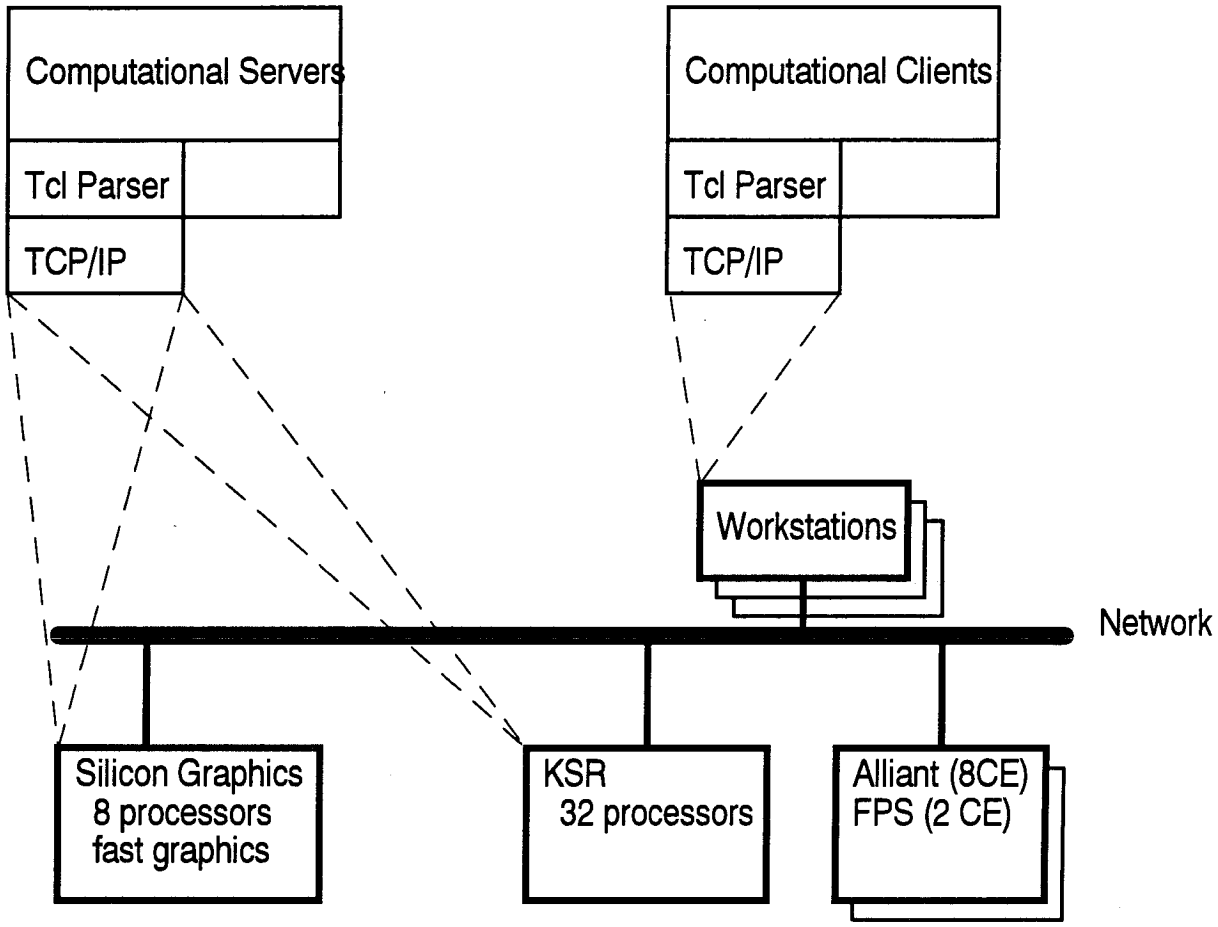


Figure 64:
Overview of software/hardware environment.
All development is currently using the Silicon Graphics with plans to move to the KSR.

6.2.5. Chemistry Server Objects

An important design choice is to decide what level of functionality in the chemistry server to bring up to the level of the interpreter; i.e., what server-specific Tcl commands should be implemented? The two possible extremes will illustrate this. At the most detailed extreme, we could implement commands for modifying every variable in the underlying C++ code. Clearly, there is no reason to go to all the trouble, and in fact it

would be downright dangerous to give clients access to internal program state variables. On the coarsest extreme we could implement one command, "Go," that would start a calculation and return the result.

Our solution is to implement what we call Tcl-enabled objects. Tcl-enabled objects are wrappers around important C++ objects in the underlying server code. Tcl-enabling C++ objects exports portions of the object functionality to the Tcl command level. The Tcl-enabled objects we have implemented are described below. By exporting important server objects to the Tcl level the user or developer can write very flexible Tcl scripts for manipulating chemically relevant objects. Later, in the case studies section, we will see how this ability allows one to prototype new algorithms using interpreted Tcl.

At the level of C++ code we have implemented a number of quantum chemistry relevant objects. For example, Molecule objects have internal state variables for describing the Gaussian basis set used in wave function calculation for the molecule. In fact, at the C++ level, GTObasisSet (Gaussian Type Orbital basis Set) objects are separate entities and Molecule objects contain an instance of a GTObasisSet. Molecule objects are Tcl-enabled, and GTObasisSet objects are not. Thus, the client to this server can manipulate Molecule objects directly, but not GTObasisSet objects. The choice of exported objects depends on what degree of control is desired from the Tcl level. In general, the Tcl level should stay at a high level of abstraction until more detail is needed.

One of the most useful Tcl-enabled objects described below is the dMatrix2 object. Clients can connect to servers with dMatrix2 objects and perform a great deal of linear algebra at the interpreted Tcl level. We have used this frequently to perform specialized manipulations using Tcl scripts. An example that will be shown is a Tcl script for diagonalizing the Fock matrix in a non-orthogonal basis set.

6.2.5.1. dMatrix2

A very useful Tcl-enabled object is the dMatrix2 object. These C++ objects describe double precision, two-dimensional matrices. A dMatrix2 object named "A" would be created in a server with the following command:

```
new dMatrix2 A
```

The following are the Tcl commands currently available for matrices. The list is as it appears in the actual C++ code. Each piece of quoted text is the Tcl command clients can use with dMatrix2 objects. The unquoted word to the right is the actual C++ function invoked in the referenced object.

```
static TclObject_Cmd commandList[] = {
  {"isSymmetric", TcldMatrix2SymmetryTest},
  {"==", TcldMatrix2EqualityTest},
  {"!=", TcldMatrix2EqualityTest},
  {"=", TcldMatrix2DupCopy},
  {"*=", TcldMatrix2SelfMult},
  {"mult", TcldMatrix2Mult},
  {"init", TcldMatrix2Init},
  {"subMatrix", TcldMatrix2SubMatrix},
  {"transpose", TcldMatrix2Transpose},
  {"eigen", TcldMatrix2Eigen},
  {"pow", TcldMatrix2Pow},
  {"invert", TcldMatrix2Invert},
  {"unit", TcldMatrix2Unit},
  {"zero", TcldMatrix2Zero},
  {NULL, NULL}
};
```

We will not describe all commands here, but the implementation of the computationally intensive functions (eigenvalue/vector solvers, inverses, matrix multiplies) demonstrates an important feature of the Tcl-enabled C++ objects. If "A" and "B" are dMatrix2 objects, then the command "`A *= B`" in a Tcl script will right-multiply A by B and place the result back in A. The command "`U eigen E`" computes the eigenvalues and eigenvectors of "U" and places the result back in U. The vector of eigenvalues is stored in "E." The key point here is that the syntax for performing these operations is very

much simplified and similar to what you might write on paper. Gone are complicated calling sequences that depend on the size of the matrices, etc.

A goal of object oriented languages is to hide the underlying *representation* of an object from the user. There are two primary advantages of this. The first is that the resulting code is easier to read: "`A *= B`" is easier to follow in source code than:[72]

```
int transa = 'N';
int transb = 'N';

double alpha = 1.0;
double beta = 0.0;

DGEMM(&transa, &transb, &rowsA, &colsB, &colsA, &alpha,
      a, &ldA, b, &ldB, &beta, ab, &ldAB);
```

As a result, it is also easier to check for correctness.

The second advantage is that high-level code of the application is independent of the underlying representation. In our specific example of a matrix multiply, we have chosen to use the LAPACK[73] routines. These routines have been optimized by experts in computational linear algebra to run on parallel processors. There obviously is no need for a computational chemist to reinvent this technology given the superior performance and reliability expected from the publicly available LAPACK routines. However, if a better set of routines is needed, or a special purpose multiplier is useful, the high-level source code

---

[72]DGEMM is the LAPACK Double precision GEneral Matrix Multiply. "`a`" and "`b`" are the matrices to be multiplied, "`ab`" is the result. "`transa`" and "`transb`" indicate if transposes are to be taken before multiplication.

[73]LAPACK, Jack Donguerra *et al.* For more information online:
For information about netlib/lapack/xnetlib
send the following message to netlib@ornl.gov
send index
send index from lapack
send index from xnetlib

containing the science need not change - only the implementation of the matrix multiply at the lowest level need be modified.

This implementation hiding is possible because the variables denoting matrices in the source code (C++ code or Tcl scripts) represent more than simple scalars - they are "objects." As such they have a set of internal state variables and a set of operations available to them. Upon instantiation of a particular operation in high-level code (like matrix multiplication), the matrix objects examine their internal state variables and determine how best to carry out the requested function. For example, in addition to the matrix elements themselves, the internal state of a dMatrix2 object contains the number of rows and columns of the stored matrix. Thus, a check is done for incompatible matrix multiplies and if the columns of the left matrix do not equal the rows of the right matrix, an error is generated. Additional state information (not present in our current implementation) might include a flag for tridiagonal matrices. In the case of a multiplication request for tridiagonal matrices, a faster routine might be called.

Comparable non-object oriented approaches to the same problem would involve several lines of code to determine the nature of the matrices (from a state variable held elsewhere in the program), and then a call the correctly named subroutine for multiplication. Two factors work against correct code in this case. First, there is *more* code and it is harder to read; second the state information is *separate* from the matrix information and thus subject to corruption elsewhere in the code.

Language Note 1:

> Complexity is virtually impossible to hide in a language without data structures (like FORTRAN 77); it may by dealt with reasonably in, for example C. However, object oriented languages like C++ make it *easy* to create source code that reflects the underlying science or mathematics.

6.2.5.2. Molecule

The Molecule object is a central object in our server architecture. Most computational chemistry servers will link in the C++ code for describing Molecules. This object is currently outfitted with the members for a quantum description of molecules. The C++ object which provides the names for Tcl-enabled Molecule object commands is shown here. A few of the commands are described.

```
   static TclObject_Cmd commandList[] = {
     {"loadIntgenFile", loadIntgenFile},
     {"loadOrbitalCardFile", loadOrbitalCardFile},
     {"loadTrnFile", loadTrnFile},
     {"deleteAtom", deleteAtom},
     {"orbitalsTodMatrix2", orbitalsTodMatrix2},
     {"basisOverlapTodMatrix2", basisOverlapTodMatrix2},
     {"orbitalEnergiesTodMatrix2", orbitalEnergiesTodMatrix2},
     {"listOrbitalEnergiesAndOccupations",
listOrbitalEnergiesAndOccupations},
     {"listMolecule", listMolecule},
     {"alignMoleculeToGrid", alignMoleculeToGrid},
     {"alignMoleculeToCrystal", alignMoleculeToCrystal},
     {"translate", translate},
     {"calculatePrimitiveAmplitudes", calculatePrimitiveAmplitudes},
     {"calculateOrbitalAmplitudes", calculateOrbitalAmplitudes},
     {"calculateLocalIonizationPotentials",
calculateLocalIonizationPotentials},
     {"calculateTotalDensity", calculateTotalDensity},
     {"calculateCurrent", calculateCurrent},
     {NULL, NULL}
   };
```

Readily identifiable are the commands for calculating various orbital amplitude related properties of the molecule. Also, note the few commands ending in "todMatrix2." These are used to transfer things like the basis set overlap matrix and the orbital coefficient matrix into dMatrix2 Tcl-enabled objects. This allows linear algebra on these matrices to be performed from a Tcl script.

6.2.5.3. ResCalc

ResCalc objects organize the process of computing the Hamiltonian matrix elements between two non-orthogonal wave functions, $\langle \Psi_A | H | \Psi_B \rangle$. These are discussed more in the case study on developing the resonance matrix element program, ResGVB.

6.2.5.4. Grid3

Grid3 objects store the information specifying a three-dimensional, rectilinear grid. Grids are used in conjunction with Molecule objects to compute various properties of the Molecule. The following are the functions known to Grid3 objects:

```
static TclObject_Cmd commandList[] = {
  {"axis", uGridAxis},
  {"ticks", uGridTicks},
  {"extents", uGridExtents},
  {"integrateScalar", uGridIntegrateScalar},
  {"squareScalar", uGridSquareScalar},
  {"filterScalar", uGridFilterScalar},
  {NULL, NULL}
};
```

The first three commands allow specification of the origin and *x,y,z*-axis direction, the number of grid elements (ticks) in each direction, and the physical-space extents along each grid axis. Implicit with the use of a grid is the calculation of a quantity on this grid. The underlying C++ objects allow for calculations of scalar and 3-vector quantities at each location of the grid. Of the Tcl commands implemented on top of the C++ objects, the last three are utility functions for manipulating scalar results.

6.2.5.5. Ints1e2e

This object type[74] holds information about the one and two-electron basis function integrals needed in quantum chemistry calculations. At present, these objects have been

---

[74]The Ints1e2e object type was designed by Terry Coley and Erik Bierwagon. The implementation was done by Erik Bierwagon.

used in conjunction with ResCalc objects to compute resonance energies. A unique feature of the methods underlying this object type is that the two electron integrals $(ij|kl)$ stored on disk can be accessed randomly and in parallel. The original programs which created old-style two electron integral files store the integrals as sequential records, each one containing all *kl*s for a particular *ij*. The integral records can only be retrieved sequentially and this restricts the algorithm one can use when calculating an energy expression. Integral storage in these old programs was optimized for calculating the Hartree-Fock or GVB energy on a single processor computer. On the other hand, our new implementation opens the way to parallel computations of energy expressions. Multiple processors can be calculating different parts of the energy expression by accessing different parts of the integral file.

The point should be made that there is no new technology here for the production of two-electron integrals. Rather what we have accomplished is two-fold: first, we have created an efficient way to use currently available two-electron integral information on modern parallel processors. Secondly, we have abstracted the methods for two-electron integral usage into an object class whose underlying implementation can be changed later to use better two-electron integral techniques. To give a specific example, the ResGVB program we have implemented will work even when we change the methods underlying the Ints1e2e object to use modern algorithms and direct (no disk storage) calculation.

6.2.5.6. PairInteraction

This object type was designed to organize the information pertaining to a pairwise interaction between atoms. These objects allowed us to write Tcl scripts to generate various super Fock matrices from the results of cluster calculations (see earlier in this thesis).

To summarize, this object contains an overlap matrix between the basis functions on each atomic center and an operator matrix that contains the matrix representation of an operator acting on the basis functions of the two atoms in the pairwise interaction. We used this object to store the Fock matrix elements of the basis functions belonging to two atomic centers. The Fock matrix elements come from a cluster calculation. Both of these matrices (overlap and operator) are rotated so that the coordinate system in which the matrices are expressed is the bond-axis coordinate system. The PairInteraction object also contains information that specifies what neighbors must exist in a target Molecule or Crystal in order for the information in this PairInteraction to be used when constructing a super-Fock matrix in a large Crystal or Molecule.

It remains to be seen whether this type of object is sufficiently general to be of widespread use. Although we have been focusing on quantum applications, it is conceivable that it would be useful to augment this type of object to optionally contain atom pairwise interaction information for other type of calculation too. For example, these objects might be useful in molecular dynamics calculations.

6.2.5.7. SelfInteraction

This object type is almost identical to the PairInteraction object type except that the overlap and operator matrices describe the self-interaction of an atom in a cluster calculation. For example, the Fock and overlap matrix elements with respect to the basis functions on a single Ni atom at the center of a Ni cluster might be stored in a SelfInteraction object and then transferred to a larger Crystal or Molecule for an approximate treatment of the larger system.

6.2.5.8. Crystal

We have begun work on a Crystal object that makes working with periodic structures easy. The following is an example from currently working code that shows an initialization block for a new Crystal object. This command would be executed in a server linked with the C++ code describing Crystal objects.

```
new Crystal X {

    spaceGroup FCC;
    latticeParameters 6.6464512 6.6464512 6.6464512;

    set sqrt2 [sqrt 2];
    set recip [expr 1.0/$sqrt2];

    unitCellTranslation X $sqrt2 0 0;
    unitCellTranslation Y 0 $recip 0;
    unitCellTranslation Z 0 0 1;

    atomUnitCellCoordinates 0 0 0 0;
    atomUnitCellCoordinates 1 $recip 0 0;
    atomUnitCellCoordinates 2 [expr 0.5*$recip] \
                              [expr 0.5*$recip] 0.5;
    atomUnitCellCoordinates 3 [expr 1.5*$recip] \
                              [expr 0.5*$recip] 0.5;

    atom Ni28 N000;
    dimensions 2 3 4;
    boundaryConditions finite finite finite
    setupNeighborLists;
}
```

The details of this are not important, however, one can glean some of the members of Crystal objects by inspecting this initialization block. The script also demonstrates the use of Tcl to facilitate parts of the initialization. In particular, two variables are defined: `sqrt2` and `recip`. These are used by the script writer to aid in specifying the locations of the atoms in the unit cell being defined. Thus, the locations of the four atoms in the FCC unit cell being defined are (0,0,0), ($\sqrt{2}/2$, 0, 0), ($\sqrt{2}/4$, $\sqrt{2}/4$, 1/2), ($3\sqrt{2}/4$, $\sqrt{2}/4$, 1/2). Our Crystal object allows the four FCC unit cell atoms to be defined in any manor. This is useful for constructing crystal fragments with particular crystal surface planes.

This choice of unit cell results in a (110) surface of the FCC lattice being generated when cells are translated in the x and y directions.

6.2.6. A Computational Services Database

We currently lack a convenient way of starting servers. Presently, the client must have a detailed knowledge of what server will perform the functions requested. Furthermore, it needs to know the exact location of the executable that must be run to start the server.

Work has begun to abstract this process. When a client needs a particular service, say wave function amplitudes, it should be able to make a request for that generic class of service from a database. The server database will maintain the information on matching services to the currently best suited server. Our initial implementation is planned around the UNIX `inetd`[75] program which will listen for connections on a well-known port number. All computational chemistry clients would request a connection to this port. Having established a connection, they would then request a service. The chemistry services program would then instantiate the correct server and establish a Tcl TCP/IP connection between the client and the server.

Planning is still in progress for this part of the system. One powerful aspect of abstracting of chemistry service requests to the database server is that the server can transparently control the resources granted. For example, the database server can automatically give the client access to the most up to date version of a chemistry server without any changes needed on the client side. Also, the database server might dynamically determine what computer to run the server on: if the server requires parallel

---

[75]The UNIX services database is a standard part of UNIX and is documented in online manual pages.

processing, the appropriate host would be chosen; if the load is too great on one machine, another can be dynamically picked, etc.

6.3. Case Studies

The first two goals we stated for our software environment were to leave the Debug-Edit-Compile-Link cycle as much as possible, and to abstract to a higher level the process of creating a computational chemistry program. At present, we have focused on achieving these goals in the area of quantum chemistry. In many respects, this is a more challenging goal than for some other areas of computational chemistry because of the complexity of the underlying methods and especially because of the need for efficient code. What will show here are some examples indicating the extent to which we have achieved these goals.

We have already seen some examples of a movement to higher level programming in our use of the features of object oriented programming which allow more robust and maintainable code to be written. (We discussed this in the context of our dMatrix2 object implementation.) We have also discussed how the underlying C++ objects are accessed in a computational server via Tcl scripts. The Tcl-enabling of C++ objects is the vehicle for exposing the high-level objects in our computational chemistry servers to the user directly (via scripts) and to client programs.

Judgment comes into play when deciding what level of abstraction to move out of statically compiled and linked code and into dynamically interpreted Tcl scripts. *Objects that are exported to the Tcl level should be of sufficient generality that the server can be made to perform new tasks not envisioned by the original programmer of the server.* As evidenced by the list of Tcl-enabled C++ objects previously described, the lowest level exported object is the double precision two-dimensional matrix. It was judged that

matrices were sufficiently useful that they should be manipulated interactively at the Tcl script level.

6.3.1. Tcl Scripts for Algorithm Prototyping

As a specific and practical demonstration of how we can reduce time spent in the DECL cycle, we present some code used to prototype image filtering algorithms for processing our infinite resolution STM images into finite resolution images. To perform the filtering, we assume that a Grid3 object named `MyGrid` has been created in the server and that some scalar function has been computed for each point on the grid. The `filterScalar` member of Grid3 objects takes one argument. The argument is a *block of Tcl code to be interpreted* for each grid point. At each grid point, `filterScalar` defines the Tcl variables `ntick`, `itick`, `location`, and `value` in the interpreter belonging to the Grid3 object. `ntick` has the Tcl list form `{ Nx Ny Nz }` where the list elements are the total number of grid points in the corresponding direction. `ilist` contains a list of three elements also, this time the indices of the current grid point. `location` is a list of three floating point values containing the real space coordinates of the grid point. Finally, `value` contains the value of the scalar function that has been evaluated previously for this grid point. Then, the code block is interpreted. The code block is a complete piece of Tcl code for performing the 25-point filter described earlier in this thesis.

```
MyGrid filterScalar {
    proc distanceSq { atom0 atom1 } {

        set distance 0.0
        for {set i 0} {$i < 3} {incr i} {
            set c0 [lindex $atom0 $i]
            set c1 [lindex $atom1 $i]
            set distance [expr {$distance + ($c1 - $c0)*($c1 - $c0)}]
        }
    return $distance
    }
```

```
    set i [lindex $itick 0]
    set j [lindex $itick 1]
    set k [lindex $itick 2]
    set nx [lindex $ntick 0]
    set ny [lindex $ntick 1]

    set result $value
    if {$i>1 && $j>1 && $i<$nx-2 && $j<$ny-2} {

        set result 0.0
        for {set i2 [expr $i-2]} {$i2<=$i+2} {incr i2} {
            for {set j2 [expr $j-2]} {$j2<=$j+2} {incr j2} {

                set cstr "{ \"%f \" coords $i2 $j2 $k}"
                set c [get NULL $cstr]
                set c [eval "list $c"]

                set d [sqrt [distanceSq $location $c]]
                set vstr "{%.20g scalar $i2 $j2 $k}"
                set v [get NULL $vstr]
                set result [expr {$result + $v*[exp -$d]}]
            }
        }
    }
    set result
}
```

We will not go through this code line by line, but once Tcl syntax is learned it is easy to generate prototype algorithms. What we have done is to "teach" the server how to do a filter without any modification to the server itself. The process of debugging this Tcl code is extremely rapid because if an error is made, the script is edited and resent to the server. There is no compile or link step and no interruption of the developer's train of thought.

Generally we will want to experiment with various algorithms. The flexibility of the server gives us that freedom. If a particular algorithm is found to be of general use, the underlying server code can be changed to provide a speed optimized version.

6.3.2. Tcl Scripts for New File Formats Without Relinking

Each of our Tcl-enabled C++ objects is outfitted with a means of obtaining the values of important state information in the object. For example, to get the number of

basis functions followed by the number of orbitals in a Molecule object using Tcl, one would write

```
Ni28 get NULL {{%d nbf} {%d norb}}
```

The keywords `nbf` and `norb` request the number of basis functions and number of orbitals. (The somewhat obscure choice of abbreviated names for these pieces of information is a hold-over from older code - better names are used in general.) The `%d` is a standard C language formatting string. The results of the each query to the object are returned using the specified format. We will see the importance of this in a moment. The `NULL` argument indicates that the result of the query should be returned to the client. This argument can also take a handle to a disk file that has been opened previously (more below). Finally, `Ni28` is the name of a Molecule object created somewhere else, and `get` is a generic command implemented in all Tcl-enabled objects to provide queries as to the object's internal state.

One application for using queries to the object is the generation of specific file formats for reading by other programs. We have implemented three such Tcl scripts. One script teaches a chemistry server to create files from Grid3 object data in the format required by a graphics program. This is how most of the two-dimensional plots in this thesis were produced. The second script outputs Grid3 object data to a file format suitable for volumetric viewing in AVS. This is how all of the three-dimensional images in this thesis were produced. The last example we have implemented outputs files from Molecule object data in the format used by the BioGraf program.

We gave specific examples to demonstrate the utility of our approach. But the most important point is that each of these was developed outside the DECL cycle and without modification to the server. Development was extremely rapid and involved a few passes in the script text editor followed by examination of the output files for correctness.

Furthermore, we have shown how a user can augment the functionality of the chemistry server without needing the source code to the server. Programming this way solves one of the most frustrating aspects of using public domain software: the file format desired is most often not supported. If the program is developed like our new servers, this is no longer a serious issue.

Another very important aspect of the way we can query objects for state information is that it opens the way to what are called "persistent objects." Put simply, many good object oriented software packages provide a method in every object for writing itself to a disk file and then recovering itself into program memory using that disk file. The process is analogous to but more general than traditional file input/output. The reason that it is more general is that since objects can be composed of sub-objects (inheritance in object oriented parlance), if each object has a method for persistent storage, all derived objects will have the same functionality with the only additional programming work being in the code to store the new pieces of the derived object.

Although we have not yet exploited it, our Tcl method for accessing object state information allows an even more general and flexible implementation of persistent objects. With little effort, and still outside the DECL cycle, we could write scripts that store and recover the state of objects. The output of the script would be another Tcl script that tells a computational server how to reconstruct the object in memory. As a result, the output object would frequently be human readable and modifiable. An additional advantage is that only selected object state need be stored. For example, the Molecule object has data for holding the basis set overlap matrix. However, instead of storing this on disk, a better approach would be to write a script that saves the basis information of the Molecule along with the Molecule object command to regenerate the overlap matrix upon reading the

stored script. Thus, we effectively store the state of the object but in a much more compact form.

6.3.3. Tcl Scripts for High-Level Computational Chemistry

As an example of allowing the user to script new computational functionality, we implemented diagonalization of the Hartree-Fock Fock matrix at the Tcl level. That is, we would like to find the eigenvalues and eigenvectors of the Fock matrix, **F**, expressed in a non-orthogonal basis set with overlap matrix, **S**. The equation is **FC** = **SCE** where we seek matrix **C** and diagonal matrix **E**. An outline of a procedure for solving the equation is given in Szabo[6] page 142. Following the text and using symmetric orthogonalization of the basis set, we wrote the following short Tcl script:

```
proc solveFC=SCE { server F S U E
                   {solver symmetric} {values_only {}} } {

    # On input, F contains the Fock matrix,
    # S contains the overlap matrix.
    # Solve FC = SCe and place the eigenvectors in the
    # matrix specified by U and the eigenvalues in the
    # matrix (vector) specified by E

    # Other input:  "server" specifies the server connection
    # to communicate with.
    # "solver" allows the caller to override the default
    # LAPACK eigensolver used.  By default we use the routine
    # for symmetric matrices, since F and S should be symmetric.
    # Finally, if "values_only" is set, then only the eigenvalues
    # will be computed - this is faster if the eigenvectors are
    # unneeded.

    # The matrices specified by U and E
    # are created in the server.

    # Use X = S^(-1/2) to transform to an orthogonal basis.
    # X is a temporary matrix we create.
    # The "pow" matrix function raises a matrix to a power
    # by diagonalizing the matrix, raising the diagonals to
    # the specified power, and then "undiagonalizing" using
    # the reverse transformation.  See matrix.c++ comments
    # for more details on matrix operations.

    $server send new dMatrix2 X = $S
    $server send new dMatrix2 $E
```

```
    $server send X pow -0.5 $E

    # Get the lowest eigenvalue of the overlap matrix
    # and output it so the caller can see if there are
    # problems with the basis set.

    set lowest [$server send $E get NULL {{%f element 0 0}}]
    puts stdout "lowest eigenvalue of S: $lowest"

    # Create the U matrix and set it equal to X^T F X

    set command "new dMatrix2 $U {mult {X transpose} $F}"
    $server send eval $command
    $server send $U *= X

    # convert U to eigenvectors, put eigenvalues in E

    puts stdout "doing eigensolve"
    $server send $U eigen $E $solver $values_only

    # Convert the eigenvectors back to the non-orthogonal
    # basis space (left multiply by X).

    $server send $U *= X left
}
```

As can be seen, the Tcl procedure is short and consists mostly of comments (beginning with "#"). This code was written and debugged without recompiling the server. The process was simply to try the script, edit any errors, and try again. The cycle is edit/debug and turn around time to correct code is very rapid.

6.3.4. Redevelopment of a Resonance GVB Program

Starting with the machinery previously discussed already in place except for the Ints1e2e object, we undertook to redevelop the resonance GVB program. Starting with a programmer with no C or C++ experience, we designed and implemented the Ints1e2e object[74] for random access to two-electron integral files. After about three months of part-time work, the programmer was up to speed on C and parts of C++ and had created the integral file reader. Simultaneously, another programmer wrote a biorthogonalization routine using LAPACK (a few function calls). Using the Ints1e2e object's capabilities, and the rest of the Molecule object functionality, it took one day to implement the main

algorithm for generating resonance matrix elements for closed shell systems. The C++ code is about 30 lines long and gave correct results on the first try after compilation.

The high degree of reusable machinery enabled a new method to be coded very quickly. All the focus could be put on the algebra of the energy expression while the details of collecting the basis set information had already been taken care of. The code reads much like you would expect from writing out the energy evaluation.

At the same time, another programmer developed, using Tk,[67] a graphical menu client to run the server and another programmer created an AVS module to control the program. We have run the resonance GVB server from Tcl scripts as well as the graphical menu. Experience suggests that three months is a very rapid turn around time for a working computational chemistry program with this level of functionality, especially using programmers with no previous experience in the languages involved.[76]

---

[76]The team consisted of Erik Bierwagon (inegrals), Matt Carlson (Tk), Ersan Demiralp (biorthogonalization), Janice Peters (orbital coefficient file reader), and Johanna Neaderhouser (AVS interface).

7. Computation of Orbital Amplitudes

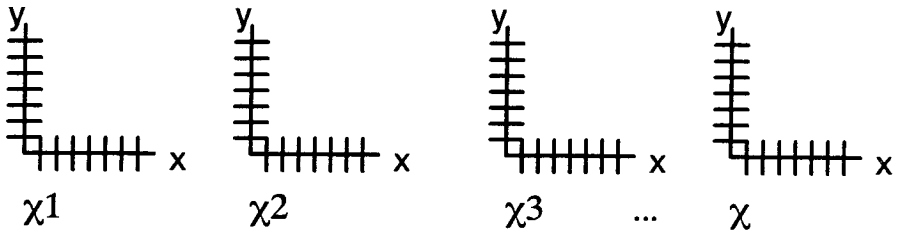
In order to compute orbital amplitudes and related quantities (e.g., local ionization potentials) rapidly for large, volumetric grids, we needed to revise the algorithms currently in use. The methods we present here are of general use because they allow rotation of the underlying basis set to any computationally convenient coordinate system. As a concrete example of how this is useful to algorithms other than orbital amplitude generation, we found these tools useful for rotating an STM model system to a coordinate system where a plane of constant *z*-value could be used as the integration plane. This greatly simplifies coding of the Bardeen tunneling matrix elements.

7.1. Old Method

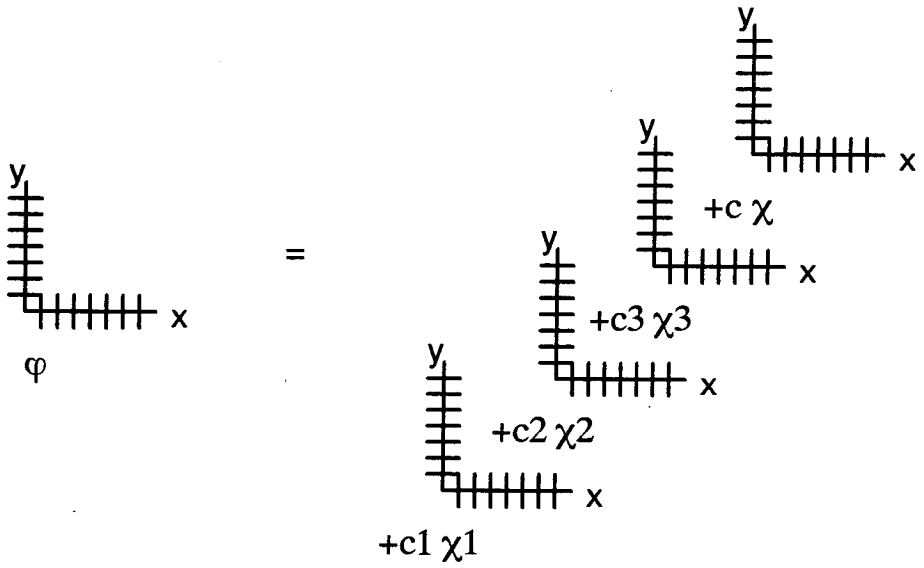
An algorithm in common use for computing orbital amplitudes and related quantities uses the following procedure.

7.1.1. Algorithm

1) Compute the amplitudes of each contracted basis function, $\{\chi_i\}$, at every location on the grid, storing the results in separate grid arrays for each basis function. Schematically, the amplitude storage for a two-dimensional grid would look like this:



2) For each grid point take the dot product of the orbital amplitudes at that grid point with the orbital coefficients. Schematically,



The example shown here is for a two-dimensional grid. However, the algorithm works analogously for three-dimensional grids; that is, a separate copy of the full three-dimensional grid is kept for each basis function to store that basis function's amplitudes.

7.1.2. Operation Count and Storage Issues

The operation count for this procedure on a regular *n*-dimensional grid scales as $O(N_{grid}^{n} N_{basisfunctions} N_{orbitals})$. Because the amplitudes are generated at every grid point for every basis function, arbitrarily oriented grids may be used - all that is required is the location of the grid points, the basis set, and the molecular coordinates.

The following table illustrates the storage problems that arise with this method. The memory requirements are calculated assuming eight byte floating point values. This method requires a separate grid storage for each basis function and for each orbital. For example, memory for the first case is calculated as 50*50*8*(100+10) bytes and divided by 2^{20} to obtain megabytes (Mb). The orbital storage figure is the amount of disk space needed to store the final orbital amplitudes on disk; the basis storage is the amount of disk space needed to store the basis function amplitudes on disk for use in rapid recalculation

of the orbital amplitudes at a later date, or for new orbital coefficients. All storage figures are in megabytes.

Case Description	Memory (Mb)	Orbital Storage	Basis Storage
50 square grid, 100 basis functions, 10 orbitals	2.1	0.2	1.9
50 square grid, 500 basis functions, 20 orbitals	9.9	0.4	9.5
50 cube grid, 100 basis functions, 10 orbitals	104.9	9.5	95.3
50 cube grid, 500 basis functions, 20 orbitals	495.9	19.1	476.8

For planar grids, this method is adequate in terms of storage requirements. However, for volumetric viewing of orbital amplitudes, the method becomes prohibitively expensive for small cases and impractical for more realistic cases. The large operation count also leads to large CPU times. (Recall that exponentials for each term in the basis function contraction are computed for each contracted basis function at each grid point.)

7.2. New Method

We have developed the following method for calculating orbital amplitudes and other properties of Gaussian wave functions on an arbitrarily oriented Cartesian grid. The method makes use of the factorizability of Gaussian basis functions to reduce the number of exponentials that must be evaluated from $O(n^3)$ to $O(n)$. The reduced number of exponential evaluations is the source of the substantial CPU time savings. The factorization of the amplitudes on the grid is the source of the substantial storage savings.

The ability to use an arbitrarily oriented, rectilinear grid (rather than one aligned with the basis set of the original calculation) is useful since we often prefer to choose an orienting plane that contains three or more atoms in the molecule. What happens in,

above, and below this plane is often of interest for studying bonding between atoms in the plane. Since the grid is finite and relatively coarse, alignment of the grid is useful for obtaining the best images. Although other programmers have used the factorability of the Gaussian basis functions to improve speed and reduce storage,[77] to our knowledge none have maintained the ability to choose a grid orientation.

7.2.1. Algorithm

1) Rotate each contracted basis function about its center such that the Cartesian coordinates of the rotated basis functions are parallel to the (orthogonal but arbitrarily oriented) axes of the grid on which the amplitudes are desired. To illustrate, let references to the original basis set have 0 superscripts. We will find a new set of orbital coefficients and basis functions to describe the same orbitals:

$$\varphi = \sum_{i=1}^{nbasisfunctions} c_i \chi_i = \sum_{i=1}^{nbasisfunctions} c_i^0 \chi_i^0 \tag{7.1}$$

2) Expand the orbital coefficient vectors to include separate values for each *primitive* basis function as in Eq. (0.24) to get

$$\varphi = \sum_{i=1}^{allprimitives} a_i G_i = \sum_{i=1}^{allprimitives} a_i N_i g_i \tag{7.2}$$

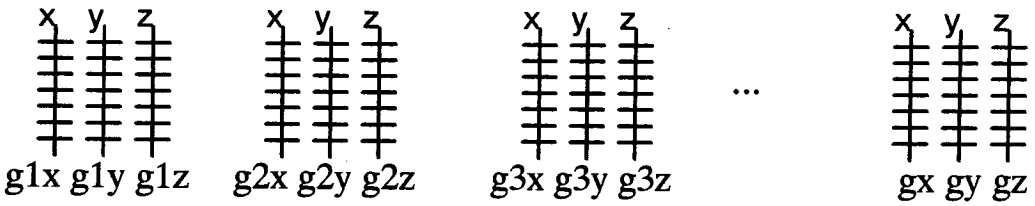
3) Factor the rotated primitive basis functions:

$$\begin{aligned} g(A,\alpha,l,m,n) &= x_A^l y_A^m z_A^n e^{-\alpha r_A^2} \\ &= (x_A^l e^{-\alpha x_A^2})(y_A^m e^{-\alpha y_A^2})(z_A^n e^{-\alpha z_A^2}) \\ &= g_x g_y g_z \end{aligned} \tag{7.3}$$

---

[77]Benzel, M., Molecular Simulations, Inc., Waltham MA, personal communication. Mark has a good implementation of fast amplitude generation.

4) Compute amplitudes of each rotated, factored primitive basis function along the edges of the grid. For a three-dimensional grid, storage of the amplitudes would look like this:



5) Now the amplitude of a primitive basis function at any point on the grid may be calculated by multiplying the factored amplitudes at the corresponding positions along each edge. For each grid position, take the dot product of the basis function amplitudes at that grid point with the orbital primitive coefficients:

$$\varphi(x,y,z) = \sum_{i=1}^{allprimitives} a_i N_i g_{x_i}(x) g_{y_i}(y) g_{z_i}(z) \qquad (7.4)$$

7.2.2. Operation Count and Storage Issues

The CPU time savings of this method derives from the fact that the number of exponentials to be calculated has been reduced to $nN_{grid}N_{allprimitives}$ for a regular n-dimensional grid. To arrive at the final orbital amplitudes at each grid point still requires $O(N_{grid}^{n}N_{allprimitives}N_{orbitals})$ operations but these are now just a few multiplies per grid point. Not needing to compute exponentials at each grid point for each basis function results in a very large reduction in CPU time. Of equal or greater significance is the reduction in the amount of storage required for the intermediate basis amplitudes. The table below shows the storage requirements for the same cases described for the old method. Calculation of the orbital storage is the same as for the old method since the orbital amplitudes still must be stored for each grid point. The basis amplitude storage, however, is computed as

50*2*8*500 bytes, using the first case as an example. The number 2 is the dimension of the grid.

Case Description	Memory (Mb)	Orbital Storage	Basis Storage
50 square grid, 500 primitives, 10 orbitals	0.6	0.2	0.4
50 square grid, 2500 primitives, 20 orbitals	2.3	0.4	1.9
50 cube grid, 500 primitives, 10 orbitals	10.1	9.5	0.6
50 cube grid, 2500 primitives, 20 orbitals	21.9	19.1	2.9

Obviously, storage needs for the basis set amplitudes have become minimal. The operation count (hence the CPU time) has also been drastically reduced. In this method exponentials for the primitives need only be computed along the edges of the grid. This comparison does not include the extra time needed to perform the basis set rotation (see Appendix), however, this is small and scales as $O(N_{basisfunctions})$. The comparison does take into account the need to deal with the basis set in terms of the larger number of primitives instead of contracted basis functions. This prefactor is estimated at 5 but would naturally be higher for cases dominated by transition metals. Again, this has no effect on the overall scaling of the number of exponentials to be computed and does not raise storage or CPU time requirements to anywhere near those of the old method when comparing volumetric computation of orbital amplitudes. Finally, we will note below that in practice it is no longer necessary to store basis set amplitudes on disk for later use since this part of the calculation has become inexpensive.

7.3. Software Design

Now we discuss the computer implementation of orbital coefficient rotation matrices. Because of the generality of our approach, this discussion addresses the broader issues of how to effectively do chemistry on the computer. We believe that the understanding and development of good technique on the computer is as essential to the effectiveness of the computational chemist as good lab technique is to the experimentalist.

7.3.1. Design Goals

In this section we focus on two design goals that guided the choice of implementation. These are general goals: the basis set rotation provides a concrete example of their efficacy in practice.

1) The parameters of the problem at hand should dictate the use of computer memory. In particular, there should be no practical limit on the angular momentum of the basis functions to be rotated.

2) The code should be as reusable as possible.

To achieve these goals we will examine two possible approaches. Equation (0.23) indicates the route to follow to generate the rotation matrices for a particular set of *l,m,n* resulting in equations like (0.26). As we see from Table 65, the size of the rotation matrices varies with the type of basis function to be rotated.

Language Note 2:

To allow space for variable size matrices we should choose a language with the ability to dynamically acquire memory from the system. FORTRAN 90, C, and C++ are just a few examples. FORTRAN 77 would not be a good choice.

7.3.2. The Direct Approach

1) Write a procedure that generates the terms for a trinomial expansion and apply this to expand the trinomials in (0.23) raised to the *l*,*m*, and *n* powers.
2) Write another procedure to multiply the three expansions together collecting like terms and ordering these in some standard order for each angular momentum (e.g.,

$$\mathbf{D} = \left[d_{x^2}, d_{y^2}, d_{z^2}, d_{xy}, d_{xz}, d_{yz}\right]$$

for a *d*-function rotation).

Step 1) is most easily accomplished by recognizing that the multinomial coefficients are generated from the following special case of Eq. (0.32):

$$\left\{\binom{p}{i,j,k} \equiv \frac{p!}{i!j!k!}\right\} \ni i+j+k=p \tag{7.5}$$

where p is the power to which the trinomial is raised and i,j,k are the powers of the first, second, and third terms in the trinomial, respectively. In order to write a general routine, it is necessary to know how many coefficients are needed for a given power, p, so that memory may be dynamically allocated according to the parameters of the problem. From Table 10, the number of coefficients is given by:

$$N_{trinomialcoefs} = \sum_{t=1}^{p+1} t = \frac{(p+1)(p+2)}{2}. \tag{7.6}$$

So, the sum of the first p+1 integers gives the length of the coefficient vector to allocate from system memory.[78]

Step 2) can be implemented as a triply-nested loop, one over the coefficients generated by the *l* trinomial expansion, one for *m*, etc. In the center of the loops a term is generated resulting from the multiplication of the current *l*,*m*, and *n* terms of the expanded

---

[78]See the Appendix for a simple way to view this which is helpful for coding.

trinomials. Examination of the summed powers of x,y,z will classify the new term as one of the possible types of functions belonging to the overall angular momentum set given by $l+m+n$. A running sum of the coefficients for each type of function in the angular momentum set is generated and the results are the values for one column of the basis function rotation matrix (0.26).

The method just described for generating a column of the basis function rotation matrix is fairly clever, will be fast, and is not practically restricted to any maximum $l+m+n$. However, it falls somewhat short on the second design goal, reusability. To the extent that we will need to generate trinomial expansions later, this code is reusable. However, we will show next that it is possible to approach the problem in a far more general way, and one that will have a much higher likelihood of being useful to another computational chemist.

7.3.3. The Symbolic Algebra Approach

The approach we have actually implemented involves writing code to treat polynomial expressions of the type in (0.23) in a general, symbolic way. That is, we will expand the expression and collect terms without relying on the fact that we are dealing with trinomials here, nor the fact that we have exactly three expansions (l,m,n) to subsequently multiply out. The ability to handle general polynomials will prove to have many advantages and additional uses.

We chose to represent the polynomial expressions as trees. Terminal "leaves" of the tree are always algebraic terms while the internal nodes represent algebraic operations. For example , the expression $(a+b+c)^l$ is represented by:

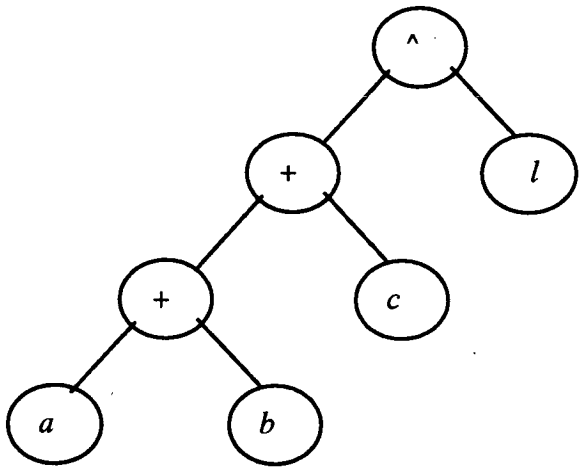


Figure 66: Expression tree for $(a+b+c)^l$

Expression trees are manipulated in two major ways: reorganizations (e.g., simplifications or expansions), and evaluations (obtaining a numerical result from the expression).

7.3.3.1. Evaluation of Expression Trees

In order to evaluate an expression a routine is written to traverse and evaluate a general expression tree. The routine takes an argument which is a pointer to the "root" of the tree, in this case the "^". When called, the evaluation routine calls itself recursively, first with a pointer to its left branch ("child") if one exists, and then with a pointer to its right subexpression. When results for the left and right branches are returned, they are combined according to the operator specified by the current node. This is called a postorder traversal of the tree[79] and is standard computer programming technique. During the recursive traversal, when an instance of the evaluation routine finds itself pointing to a node with no children, the current node is necessarily a term or "leaf." Accordingly, a

---

[79]E. M. Reingold, W. J. Hansen, **Data Structures**, Little, Brown & Co., Boston, MA (1983). ISBN 0-316-73951-0

constant for the current variable is returned to the previous level. If at any point a leaf is reached and no constant can be obtained (because a variable was not set), then the expression is undefined and this is an error condition.

A very useful feature of this representation is the simplicity of programming. The code required to evaluate an expression tree amounts to a "case" statement to determine what kind of operator or term the current node holds, and one line to perform the evaluation on the results from a recursive evaluation of the left and right side. And here is the point: due to the recursive nature of the routine, once this routine is debugged for all cases (operators) of a tree as simple as $a+b$, it can be proven correct for any more complex tree.

Language Note 3:

To allow for the elegant use of recursion we should choose a language with support for recursion. FORTRAN 90, C, and C++ are just a few examples. FORTRAN 77 does not support recursion and, therefore, would not be a good choice.

7.3.3.2. Rearrangements of Expression Trees

Besides routines to evaluate expression trees, we need routines to manipulate them. In particular, we need a routine to generate a new expression tree which is the distributed expansion of the original tree. As an example, we need to perform reconstructions like this:

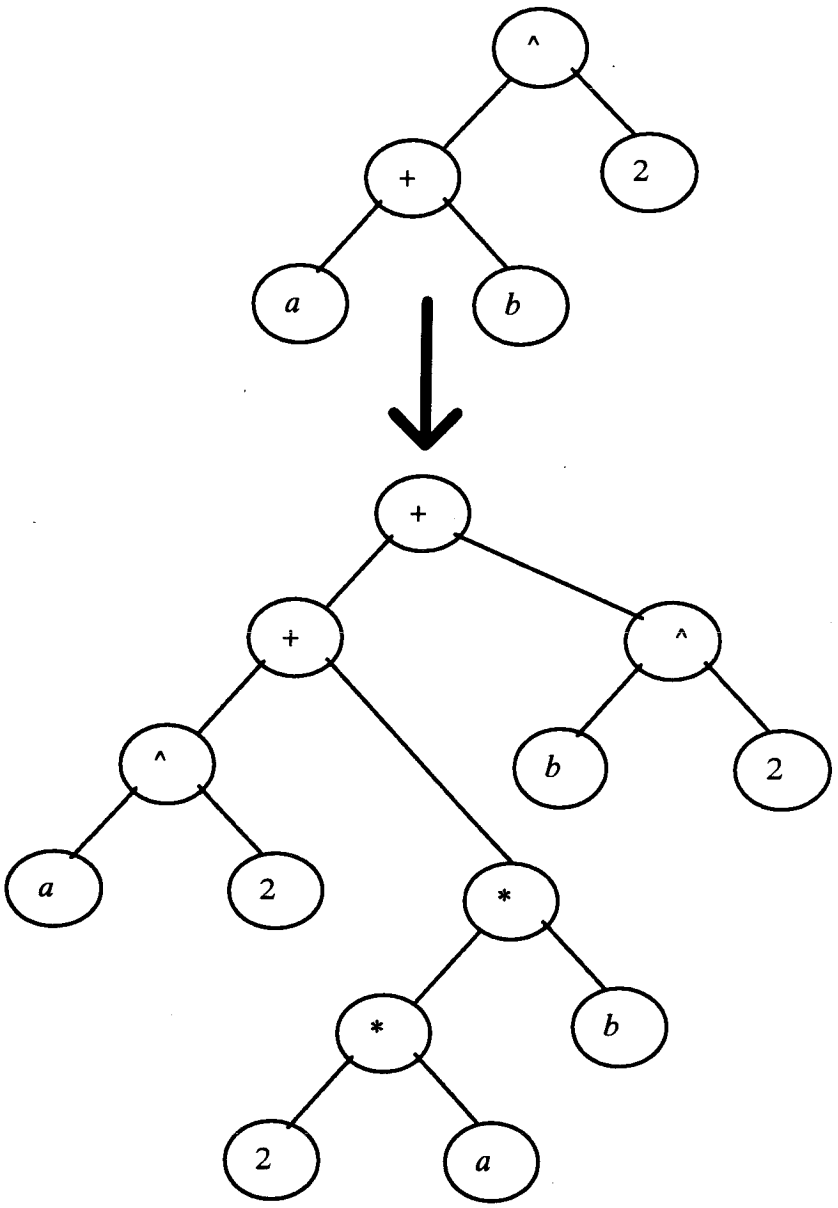


Figure 67:
Expression tree for expansion of $(a+b)^2 \rightarrow a^2 + 2ab + b^2$

The implementation details are not really important. We will just comment on a few salient points. First, applying the distributed law simply involves traversing the original tree in the correct order and applying the distributive law to each subtree. To handle the

multinomial expansions, we consider every multinomial a binomial with one lone term on the right and a sum of terms on the left. The binomial is expanded in one step using the binomial coefficients from Table 12 (Figure 67 is the example for p=2). Each resulting term is passed as a recursive call to the distribute routine until no more expansion is possible. At the end, when the initial instance of the routine has returned from all recursive calls, the final tree must be processed one more time to collect like terms.

7.3.3.3. Basis Set Rotation Matrices as Sets of Expressions

After having expanded a particular basis function from Eq. (0.23), we have an expression which can have hundreds of terms. (We have tested *i*-functions.) The terms are collected so that the expression is arranged as a series of coefficients (themselves polynomials in terms of *a,b,c...*) multiplying terms of the form $x^l y^m z^n$. These coefficient expressions are stored in a column vector of pointers to expression trees. After expansions are done for each basis function in an angular momentum set, we have a collection of columns that represents the basis function rotation matrix. For the *d*-functions this matrix looks like the rotation matrix in Eq. (0.26). In fact, for diagnostic purposes, we can simply print out the symbolic representation of our rotation matrix and compare it with what we expect from doing the algebra for low angular momentum cases. Once we are confident of the first few functions (say up to *d*-functions) we can be sure that higher angular momenta are handled correctly, just as if we had written out all the terms - but with no mistakes!

Having stored the symbolic representation of each basis function rotation matrix we need, we can evaluate the matrix for any set of angles, $\{\alpha,\beta,\gamma\}$, by assigning actual numerical values to the *a,b,c...* and running each matrix element (expression) through the expression evaluator to obtain an actual number. (We need a way to get a Cartesian rotation matrix, Eq. (0.18), for an arbitrarily oriented coordinate system; this is standard

computer graphics fare.[80]) This evaluation results in a particular basis function rotation matrix which can be passed to another routine, e.g., a FORTRAN routine, to numerically evaluate the new orbital coefficients using the rotation matrices.

7.3.3.4. Self-Augmenting Basis Sets

The advantages to this symbolic approach are numerous. We will illustrate the point with an example: it is almost always the case that one uses an angularly complete basis set in the sense that if one function of a particular angular momentum set is present, so are all the others. That is, if you use a p_x, chances are you have a p_y and a p_z also. All old quantum programs for manipulating wave functions based on Gaussian basis sets that I am aware of are written to assume this, and there is no way for the user to change this. However, there are cases where it is actually useful to restrict your basis for some part of the calculation to one component, e.g., p_z to study π-space effects. The programs we have written automatically augment the basis set during rotation where needed. This becomes such a simple addition that there is no reason not to support it.

---

[80]D. F. Rogers, **Mathematical Elements for Computer Graphics**, 2nd Ed., McGraw-Hill (1990), p. 123.

I. Appendix

This appendix presents a rapid overview of the operating modes of the STM which are referred to elsewhere in this text. For more details, the reader is referred to the vast body of literature on STM experimental designs.[81] In each case below, we assume a tunneling current has already been established. Methods for bringing the tip into tunneling range are discussed at length in the STM experimental literature. Almost all STM experiments involve raster scanning the tip over the surface to cover a specific area with measurements taken at regular intervals (pixels) during the scan to build up the image. We will generally not be concerned with the details of scanning here.

A. Overview of Some STM Operating Modes

1. Constant Current Mode

In constant current mode a bias voltage and tunneling current set point are chosen. Currents are typically in the nanoamp range. Bias voltages vary considerably depending on the electronic structure of the system and the location of the states of interest relative to the Fermi energy. The tip is then scanned over the surface. As the tunneling probability changes due to topological and electronic changes on the surface, the feedback voltage to the *z*-piezo will respond. The response will extend or retract the tip in order to re-establish tunneling current at the set point. The voltage applied to the *z*-piezo is recorded at regular pixels during the raster scan to accumulate the image.

---

[81] A good starting place is Binnig, G., Rohrer, H., *Surface Science*, **126** 236-244 (1983).

2. Constant Height Mode

The goal of constant height imaging is to acquire an image with the tip moving in some fixed plane above the surface. In reality the exact orientation and distance of the scan plane with respect to the sample is not clearly defined. Experimentally, an average current is chosen and the feedback response time for movement of the *z*-piezo is made slow compared to the motion of the *x* and *y*-piezo across the surface. The idea is to establish movement of the tip in the plane that generates some average current over the whole image. The actual image is acquired by rapidly sampling the (variable) current at each pixel of the scan.

3. Barrier Height Imaging

With a stable instrument it is possible to break the *z*-piezo feedback loop temporarily at each pixel of the image. Having done this, a voltage ramp may be applied to the *z*-piezo resulting in an extension or contraction of the tip. The tunneling current is recorded during this ramp and then the feedback loop is reestablished and the scan proceeds to the next pixel. The result of this technique is that a series of current vs. tip-sample distance curves are acquired simultaneously with and at each pixel of a standard slow scan (constant current) image. This current vs. distance information is used to calculate effective barrier heights at each pixel; see Eq. (1.2).

4. Atomic Force Microscopy

Early on in the days of STM it was found that the tip could actually be dragged over the sample surface and the deflections of the cantilever holding the tip measured with a deflected laser beam.[82] Very good instruments have been able to achieve atomic

---

[82]There are many experimental refinements for AFM. See the work of P. Hansma, for example.

resolution on flat surfaces. Somewhat more limited success has been had with adsorbate systems. In each case we expect the AFM image to be a map of the total electron density of the sample as it is traced with a finite radius probe tip.

B. Manipulations of Gaussian Basis Sets

The Hartree-Fock (HF) and Generalized Valence Bond (GVB) wave functions computed in this work are all real functions expressed as expansions in terms of a finite set of functions called the basis set. The members of the basis set are polynomial-Gaussian functions of the Cartesian coordinates. In order to achieve computational efficiencies and generality in much of our software, it has frequently been necessary to manipulate these functions explicitly. Therefore, it is useful to review and define our notation for working with Gaussian basis functions.

1. Review and Notation

We will need to refer to "primitive" basis functions and "contracted" basis functions as described below. Where we simply refer to a "basis function" we will assume this can be a contracted basis function.

a) Primitive Gaussian Basis Functions

An unnormalized Gaussian basis function is given by

$$g(x,y,z;A,\alpha,l,m,n) = x_A^l y_A^m z_A^n e^{-\alpha r_A^2} \qquad (0.1)$$

where point $A(A_x,A_y,A_z)$ is the center of the Gaussian with $x_A = x - A_x$, and similar for y and z, and $r_A^2 = x_A^2 + y_A^2 + z_A^2$. Basis functions are classified into groups by the sum of the integral polynomial exponents, $l+m+n$, and we will loosely refer to such groups as "angular momentum sets."

G to refers to normalized Gaussian functions:

$$G(x,y,z;A,\alpha,l,m,n) = N(\alpha,l,m,n)g(x,y,z) \tag{0.2}$$

where the normalization constant with respect to integration from -∞ to ∞ is given by[83]

$$N(\alpha,l,m,n) = \frac{1}{\sqrt{\langle g|g\rangle}} = \sqrt{\frac{2^{2(l+m+n)+\frac{3}{2}}\alpha^{l+m+n+\frac{3}{2}}}{(2l-1)!!(2m-1)!!(2n-1)!!\pi^{\frac{3}{2}}}}. \tag{0.3}$$

The notation "!!" indicates an odd factorial, $(2p-1)!! = 1\cdot 3\cdot 5\cdots(2p-1)$, and when $p=0$, then $(2p-1)!! \equiv 0$. Equation (0.3) is easily verified using the more commonly known one dimensional integral[84]:

$$\int_0^\infty x^{2n}e^{-ax^2}dx = \frac{1\cdot 3\cdot 5\cdots(2n-1)}{2^{n+1}a^n}\sqrt{\frac{\pi}{a}}. \tag{0.4}$$

Taking the point A to be at the origin, and recognizing that the integrand in (0.4) is symmetric around zero, the derivation is started as follows...

$$\begin{aligned}\langle g|g\rangle &\equiv \int_{-\infty}^{\infty} g^* g d\tau \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x^l y^m z^n e^{-\alpha r^2})x^l y^m z^n e^{-\alpha r^2}dxdydz \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(A,2\alpha,2l,2m,2n)dxdydz \\ &= \int_{-\infty}^{\infty}x^{2l}e^{-2\alpha x^2}dx\int_{-\infty}^{\infty}y^{2m}e^{-2\alpha y^2}dy\int_{-\infty}^{\infty}z^{2n}e^{-2\alpha z^2}dz\,.\end{aligned} \tag{0.5}$$

b) Contracted Gaussian Basis Functions

An unnormalized contracted Gaussian basis function is a fixed linear combination of *normalized* primitive Gaussian functions all on the same center:

$$\gamma(x,y,z;A,l,m,n) = \sum_{i=1}^{nprimitives} k_i G_i(x,y,z;A,\alpha_i,l,m,n). \tag{0.6}$$

---

[83]H. Taketa, S. Huzinaga, K. O-ohata, *J. Phys. Soc. Japan*, **21** 2313 (1966).

[84]Standard Mathematical Tables, 24th Ed., CRC Press, 1976, equation 666, p. 391.

The k_i and α_i are predetermined and fixed for the purposes of further computations. The normalized contracted basis function is given by

$$\chi = N\gamma \tag{0.7}$$

and the normalization constant can be determined from the expression $N = 1/\sqrt{\langle\gamma|\gamma\rangle}$ and:

$$\begin{aligned}\langle\gamma|\gamma\rangle &= \left\langle \sum_{i=1}^{nprimitives} k_i G_i \middle| \sum_{j=1}^{nprimitives} k_j G_j \right\rangle \\ &= \sum_{i,j} k_i k_j N_i N_j \langle g_i | g_j \rangle\end{aligned} \tag{0.8}$$

by recognizing that the integrals of primitive Gaussians arising in the denominator can be evaluated using (0.4) and following (0.5) since:

$$\langle g_i(A,\alpha_i,l,m,n)|g_j(A,\alpha_j,l,m,n)\rangle = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(A,\alpha_i+\alpha_j,2l,2m,2n)dxdydz. \tag{0.9}$$

c) Molecular Orbitals

Molecular orbitals are expressed as linear combinations of *normalized* contracted basis functions:

$$\varphi = \sum_{i=1}^{nbasisfunctions} c_i \chi_i \tag{0.10}$$

where the c_i are to be determined, e.g., in an SCF calculation.

d) Derivatives of Gaussian Functions

It will be useful to review the derivative properties of Gaussian functions. Recalling Eq. (0.1) we note that the primitive Gaussian function can be separated into a product of functions in x, y, or z:

$$\begin{aligned}g(x,y,z;A,\alpha,l,m,n) &= x_A^l y_A^m z_A^n e^{-\alpha x_A^2} e^{-\alpha y_A^2} e^{-\alpha z_A^2} \\ &= (x-A_x)^l (y-A_y)^m (z-A_z)^n e^{-\alpha(x-A_x)^2} e^{-\alpha(y-A_y)^2} e^{-\alpha(z-A_z)^2}.\end{aligned} \tag{0.11}$$

Consider the general operator

$$\hat{D} = \frac{\partial^p}{\partial x^p}\frac{\partial^q}{\partial y^q}\frac{\partial^r}{\partial z^r}$$

applied to (0.11). We can apply the derivatives separately. For the case $p = 1$ the x result is:

$$\frac{\partial}{\partial x} g(x,y,z;A,\alpha,l,m,n) =$$
$$\begin{cases} (-2\alpha)g(x,y,z;A,\alpha,l+1,m,n); & l = 0 \\ (-2\alpha)g(x,y,z;A,\alpha,l+1,m,n) + (l)g(x,y,z;A,\alpha,l-1,m,n); & l > 0 \end{cases} \quad (0.12)$$

Since the results of differentiation involve only Gaussians with respect to the same center shifted to one higher and one lower angular momentum exponent, it is easy to apply the general differentiation operator recursively using the same algorithm.

e) Combining Gaussians on Different Centers

The product of two primitive Gaussian basis functions centered at different points is a sum of new Gaussian functions all centered about a point between the original two points:[85]

---

[85]For a particularly clear discussion of one-electron Gaussian integrals, see J. D. Augspurger and C. E. Dykstra, *J. Comp. Chem.* **11** 105 (1990).

$$g(x,y,z;A,\alpha_A,l_A,m_A,n_A)g(x,y,z;B,\alpha_B,l_B,m_B,n_B)$$
$$= D_x D_y D_z \begin{bmatrix} (x_P + (P_x - A_x))^{l_A} (x_P + (P_x - B_x))^{l_B} \\ \cdot (y_P + (P_y - A_y))^{m_A} (y_P + (P_y - B_y))^{m_B} \\ \cdot (z_P + (P_z - A_z))^{n_A} (z_P + (P_z - B_z))^{n_B} \end{bmatrix} e^{-(\alpha_A+\alpha_B)r_P^2}$$

where

$$P_x = \frac{\alpha_A A_x + \alpha_B B_x}{\alpha_A + \alpha_B}, etc. \qquad D_x = \exp\left(\frac{-\alpha_A \alpha_B (A_x - B_x)^2}{\alpha_A + \alpha_B}\right), etc.$$

(0.13)

Since the components of P, A, and B are just constants, when the terms in the brackets are expanded, the result is a sum of new Gaussians now centered about P and with angular exponents in each coordinate ranging from the sum of the original two functions down to s-functions.

f) Some Notes on Normalization Constants

From the standpoint of programming, one of the more tedious tasks is keeping proper track of normalization constants when manipulating molecular orbitals expressed in Gaussian basis sets. Therefore, we collect here some of the more common cases of normalization bookkeeping. In general, coefficient vectors will multiply *normalized* functions.

Uncontraction of the orbital coefficient vector:

A useful routine to have will implement the following for each term in the molecular orbital expansion over normalized, contracted functions.

$$
\begin{aligned}
c\chi &= cN_c\gamma \\
&= cN_c\left(\sum_i k_i G_i\right) \\
&= \underbrace{cN_c k_1}_{a_1} G_1 + \underbrace{cN_c k_2}_{a_2} G_2 + \cdots \\
&= \sum_j a_j G_j
\end{aligned}
\tag{0.14}
$$

Expanding derivatives of the molecular orbital:

The first step is to uncontract the orbital using a routine designed to implement Eq. (0.14). The molecular orbital, then expressed in normalized primitives, is differentiated with a routine to implement the following for each term:

$$
\begin{aligned}
\frac{\partial G}{\partial x} &= \frac{\partial (Ng(x,y,z;A,\alpha,l,m,n))}{\partial x} \\
&= N(-2\alpha\, g^+ + l\, g^-) \\
&= \underbrace{\frac{-2\alpha N}{N_{g^+}}}_{a^+} G^+ + \underbrace{\frac{l N}{N_{g^-}}}_{a^-} G^-
\end{aligned}
\tag{0.15}
$$

Multiplication of molecular orbitals:

Again, the first step is to uncontract each orbital using Eq. (0.14). The subsequent multiplication of orbitals results in terms quadratic in normalized primitives. These are combined as

$$
\begin{aligned}
a_1 G_1 a_2 G_2 &= a_1 N_1 g_1 a_2 N_2 g_2 \\
&= a_1 a_2 N_1 N_2 D_x D_y D_z \sum_i g_{12}^{(i)} \\
&= \underbrace{\frac{a_1 a_2 N_1 N_2 D_x D_y D_z}{N_{12}^{(1)}}}_{b_1} G_{12}^{(1)} + \underbrace{\frac{a_1 a_2 N_1 N_2 D_x D_y D_z}{N_{12}^{(2)}}}_{b_2} G_{12}^{(2)} + \cdots \\
&= \sum_j b_j G_{12}^{(j)}
\end{aligned}
\tag{0.16}
$$

where we have used Eq. (0.13) to combine the unnormalized primitives. The unnormalized primitives in the summation result from expanding the square brackets of

Eq. (0.13); each of these must be renormalized in order to express the result in terms of normalized primitives. Again, writing a general routine to handle this produces consistent use of coefficients/basis functions throughout the code resulting in less confusion and easier modification later.

2. Rotation of the Basis Set

Often, when manipulating or analyzing HF or GVB wave functions, it is useful to re-express the orbitals in terms of Gaussian basis functions that are defined in a coordinate system different from that of the basis functions used in the original self-consistent field (SCF) calculations. In this section we outline the necessary algebra and describe the computationally general approach we have developed for rotating the basis set orientation in Cartesian space. We will note why it is difficult to create such general code in FORTRAN 77 and how these difficulties are overcome with modern languages.

a) Algebra of Basis Set Rotation

To begin we recall that to rotate a point in space we can apply a 3 by 3 rotation matrix

$$[x,y,z] = [x_0,y_0,z_0]\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}. \qquad (0.17)$$

We will sometimes refer to zero-subscripted coordinates as "world" coordinates and unsubscripted coordinates as "grid" coordinates. This wording reflects the fact that we often are transforming from an original frame of reference to a new reference frame such as a bond-axis frame or computational grid frame. The rotation matrix is generally obtained from some arbitrary concatenation of rotations about the principal axes world axes,

$$\mathbf{R} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \mathbf{R}_{x_0}\mathbf{R}_{y_0}\mathbf{R}_{z_0}\cdots \tag{0.18}$$

where,

$$\mathbf{R}_{x_0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{0.19}$$

$$\mathbf{R}_{y_0} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{0.20}$$

$$\mathbf{R}_{z_0} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{0.21}$$

For our purposes it will be convenient to look at the reverse rotation, $[x,y,z]\mathbf{R}^{-1} = [x_0,y_0,z_0]$ which shows how to express a point in the world coordinate system in terms of the grid coordinate system. Since a pure rotation matrix is an orthogonal matrix (real and unitary[86]):

$$\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}. \tag{0.22}$$

Consider rotating a Gaussian basis function, $g(x_0,y_0,z_0;A,\alpha,l,m,n)$, about its center, A. The exponential part is radially symmetric about A, but the polynomial part will be modified when expressed in the new coordinate system. For now, we will take A to be at the origin. The polynomial can be expressed in the new coordinate system by explicitly using the elements of the rotation matrix:

---

[86]J. D. Foley, A. VanDam, **Fundamentals of Interactive Computer Graphics**, Addison-Wesley (1982), p. 258. ISBN 0-201-14468-9.

$$x_0^{l_0} y_0^{m_0} z_0^{n_0} = (ax + by + cz)^{l_0} (dx + ey + fz)^{m_0} (gx + hy + iz)^{n_0}. \qquad (0.23)$$

The result of expanding this expression is a series of terms in x, y, z raised to various powers,

$$x_0^{l_0} y_0^{m_0} z_0^{n_0} = \sum_{l,m,n}^{l+m+n=l_0+m_0+n_0} c_{l,m,n} x^l y^m z^n.$$

The sum of these powers for each term is equal to $l_0 + m_0 + n_0$, and therefore the original polynomial function has been expanded in terms of basis functions in the new coordinate system belonging to the same angular momentum set as the functions in the original coordinate system. For example, a p-function ($l_0 + m_0 + n_0 = 1$) is expressed as a linear combination of p-functions in the rotated coordinate system, as expected.

b) Expressing Orbitals in the Rotated Basis Set

If we have an orbital expressed in terms of coefficients of world coordinate basis set, we can express the same orbital in terms of the rotated (grid coordinate or bond axis coordinate) basis set by finding the matrix which maps the coefficients of one basis set to those of the other using (0.23). To see this we will expand the orbital out fully in terms of coefficients on *unnormalized* primitive basis functions

$$\varphi = \sum_{i=1}^{nbasisfunctions} c_i \chi_i = \sum_{i=1}^{nbasisfunctions} c_i N_i \sum_{j=0}^{nprimitives_i} k_j^{(i)} G_j^{(i)} \equiv \sum_{k=0}^{allprimitives} a_k g_k \qquad (0.24)$$

where the a_k have incorporated the primitive Gaussian normalization constants. Later, we will show that with care the rotation can be done in terms of normalized, contracted basis functions.

As an example, we will generate the rotation matrix for the d-functions by rotating each polynomial in turn:

$$
\begin{aligned}
x_0^2 &= (ax+by+cz)^2 = a^2x^2 + b^2y^2 + c^2z^2 + 2abxy + 2acxz + 2bcyz \\
y_0^2 &= (dx+ey+fz)^2 = d^2x^2 + e^2y^2 + f^2z^2 + 2dexy + 2dfxz + 2efyz \\
z_0^2 &= (gx+hy+iz)^2 = g^2x^2 + h^2y^2 + i^2z^2 + 2ghxy + 2gixz + 2hiyz \\
x_0y_0 &= (ax+by+cz)(dx+ey+fz) \\
&= adx^2 + bey^2 + cfz^2 + (ae+bd)xy + (af+cd)xz + (bf+ce)yz \\
x_0z_0 &= (ax+by+cz)(gx+hy+iz) \\
&= agx^2 + bhy^2 + ciz^2 + (ah+bg)xy + (ai+cg)xz + (bi+ch)yz \\
y_0z_0 &= (dx+ey+fz)(gx+hy+iz) \\
&= dgx^2 + ehy^2 + fiz^2 + (dh+eg)xy + (di+fg)xz + (ei+fh)yz
\end{aligned}
$$

(0.25)

The resulting matrix equation is:

$$
\left[d_{x_0^2}, d_{y_0^2}, d_{z_0^2}, d_{x_0y_0}, d_{x_0z_0}, d_{y_0z_0}\right] =
$$

$$
\left[d_{x^2}, d_{y^2}, d_{z^2}, d_{xy}, d_{xz}, d_{yz}\right]
\begin{bmatrix}
a^2 & d^2 & g^2 & ad & ag & dg \\
b^2 & e^2 & h^2 & be & bh & eh \\
c^2 & f^2 & i^2 & cf & ci & fi \\
2ab & 2de & 2gh & (ae+bd) & (ah+bg) & (dh+eg) \\
2ac & 2df & 2gi & (af+cd) & (ai+cg) & (di+fg) \\
2bc & 2ef & 2hi & (bf+ce) & (bi+ch) & (ei+fh)
\end{bmatrix}
$$

$$
= \left[d_{x^2}, d_{y^2}, d_{z^2}, d_{xy}, d_{xz}, d_{yz}\right] \mathbf{R}_{\mathbf{d}}^{\alpha,\beta,\gamma}
$$

(0.26)

where $d_{x^2} \equiv g(x,y,z;A(0,0,0),\alpha,2,0,0)$, etc., and α, β, γ refer to the fundamental rotations used to generate the Cartesian rotation matrix (Eq. (0.18)). The part of an orbital involving a particular *d*-basis function is expressed as a dot product of the appropriate coefficients with these *d*-basis functions, or in matrix notation the row vector of *d*-functions is multiplied by the column vector of coefficients,

$$\begin{aligned}\varphi_{\{d\}} &= \mathbf{D_0 C_0} \\ &= \mathbf{D R_d^{\alpha,\beta,\gamma} C_0} \\ &= \mathbf{DC}\end{aligned} \tag{0.27}$$

where,

$$\mathbf{C_0} = \left[c_{x_0^2}, c_{y_0^2}, c_{z_0^2}, c_{x_0y_0}, c_{x_0z_0}, c_{y_0z_0}\right]^{\mathrm{T}}$$
$$\mathbf{D_0} = \left[d_{x_0^2}, d_{y_0^2}, d_{z_0^2}, d_{x_0y_0}, d_{x_0z_0}, d_{y_0z_0}\right].$$

Because it will be helpful later in conceptualizing the computer implementation, we show explicitly for the *d*-functions the form of the new coefficient vector,

$$\mathbf{R_d^{\alpha,\beta,\gamma} C_0} = \mathbf{R_d^{\alpha,\beta,\gamma}} \begin{bmatrix} c_{x_0^2} \\ c_{y_0^2} \\ c_{z_0^2} \\ c_{x_0y_0} \\ c_{x_0z_0} \\ c_{y_0z_0} \end{bmatrix} =$$

$$\begin{bmatrix} c_{x_0^2}a^2 + c_{y_0^2}d^2 + c_{z_0^2}g^2 + c_{x_0y_0}ad + c_{x_0z_0}ag + c_{y_0z_0}dg \\ c_{x_0^2}b^2 + c_{y_0^2}e^2 + c_{z_0^2}h^2 + c_{x_0y_0}be + c_{x_0z_0}bh + c_{y_0z_0}eh \\ c_{x_0^2}c^2 + c_{y_0^2}f^2 + c_{z_0^2}i^2 + c_{x_0y_0}cf + c_{x_0z_0}ci + c_{y_0z_0}fi \\ 2c_{x_0^2}ab + 2c_{y_0^2}de + 2c_{z_0^2}gh + c_{x_0y_0}(ae+bd) + c_{x_0z_0}(ah+bg) + c_{y_0z_0}(dh+eg) \\ 2c_{x_0^2}ac + 2c_{y_0^2}df + 2c_{z_0^2}gi + c_{x_0y_0}(af+cd) + c_{x_0z_0}(ai+cg) + c_{y_0z_0}(di+fg) \\ 2c_{x_0^2}bc + 2c_{y_0^2}ef + 2c_{z_0^2}hi + c_{x_0y_0}(bf+ce) + c_{x_0z_0}(bi+ch) + c_{y_0z_0}(ei+fh) \end{bmatrix}$$
$$= \mathbf{C}$$

. (0.28)

Equation (0.28) is generic for any set of primitive *d*-basis functions where a "set" means the collection of *d*-functions at a particular center that can rotate into each other. It is only necessary to construct generic rotation matrices for each angular momentum represented in the original basis set. The following table shows the relationship between

the coefficient rotation matrix size and the angular momentum of the function set to be rotated:

Function	N	Coefficient Rotation Matrix Size
S	0	1x1
P	1	3x3
D	2	6x6
F	3	10x10
	N	$\frac{(N+1)(N+2)}{2}$

Table 11:
Size of rotation matrix as a function of angular momentum.

c) Rotation of Contracted Basis Functions

As might be expected, it is possible to generate a single rotation matrix for a normalized, contracted Gaussian function. However, we must fill in a few details avoided in the discussion above. First we take a closer look at the normalized, contracted Gaussian basis function Eqs. (0.6) and (0.7) as expressed in "world" (zero subscripted) coordinates:

$$\begin{aligned}\chi_0(x_0,y_0,z_0;A_0,l_0,m_0,n_0) &= N\sum_{i=1}^{nprimitives} k_i G_i(x_0,y_0,z_0;A_0,\alpha_i,l_0,m_0,n_0)\\ &= N\sum_{i=1}^{nprimitives} k_i N_i g_i(x_0,y_0,z_0;A_0,\alpha_i,l_0,m_0,n_0)\\ &= N\sum_{i=1}^{nprimitives} k_i N_i x_{A_0}^{l_0} y_{A_0}^{m_0} z_{A_0}^{n_0} e^{-\alpha_i r_{A_0}^2}\\ &= N x_{A_0}^{l_0} y_{A_0}^{m_0} z_{A_0}^{n_0} \sum_{i=1}^{nprimitives} k_i N_i(\alpha_i,l_0,m_0,n_0) e^{-\alpha_i r_{A_0}^2}\end{aligned}$$

. (0.29)

In the last step we have emphasized that the primitive Gaussian normalization constant is a function of the radial exponent as well as the polynomial exponents. For now we take A at the origin and make the rotation substitution for $x_{A_0}^{l_0} y_{A_0}^{m_0} z_{A_0}^{n_0}$ in Eq. (0.29) indicated by Eq. (0.23):

$$\begin{aligned}&\chi_0(x_0,y_0,z_0;A_0,l_0,m_0,n_0)\\ &= N\sum_{l,m,n}^{l+m+n=l_0+m_0+n_0} c_{l,m,n} x^l y^m z^n \sum_{i=1}^{nprimitives} k_i N_i(\alpha_i,l_0,m_0,n_0) e^{-\alpha_i r_A^2}\\ &= N\sum_{l,m,n}^{l+m+n=l_0+m_0+n_0} c_{l,m,n} x^l y^m z^n \sum_{i=1}^{nprimitives} k_i \frac{N_i(\alpha_i,l_0,m_0,n_0)}{N_i(\alpha_i,l,m,n)} N_i(\alpha_i,l,m,n) e^{-\alpha_i r_A^2} \qquad (0.30)\\ &= \sum_{l,m,n}^{l+m+n=l_0+m_0+n_0} c'_{l,m,n} N x^l y^m z^n \sum_{i=1}^{nprimitives} k_i N_i(\alpha_i,l,m,n) e^{-\alpha_i r_A^2}\end{aligned}$$

in the last step we have arranged the terms to show that the right hand side can be expressed as a sum of normalized, contracted functions belonging to the same angular momentum set as the original function. The critical assumption that we have made is that the ratios of primitive Gaussian normalization constants in the contraction are equal for all α_i and can therefore be subsumed into the $c'_{l,m,n}$. That this is true can be seen from Eq. (0.3) where we see that α is raised to the sum of the polynomial exponents, and we know $l+m+n = l_0+m_0+n_0$, and this does not change during the contraction summation. Thus,

the ratios of primitive Gaussian normalization constants seen in Eq. (0.30) depend only on the ratios of the odd factorials in Eq. (0.3) and these are constant during the contraction summation also, since l, m, n, l_0, m_0, n_0 are all constant during the contraction summation.

In conclusion, this means that if we multiply the matrix elements of the polynomial function rotation matrices by the ratio of the primitive Gaussian normalization constants for the world coordinate and grid coordinate primitives, we can use the rotation matrix on normalized, contracted Gaussian basis functions with the origin at (0,0,0).

d) General Rotation of Contracted Basis Functions

In a polyatomic system we will have molecular orbitals expressed as a linear combinations of normalized, contracted basis functions at many centers. To complete our discussion of basis set rotation, we examine rotation of the coordinate system when the basis functions have arbitrary centers. Fortunately, this is a straightforward extension of (0.23):

$$
\begin{aligned}
&(x_0 - A_{x_0})^{l_0}(y_0 - A_{y_0})^{m_0}(z_0 - A_{z_0})^{n_0} \\
&= \\
&(ax + by + cz - aA_x - bA_y - cA_z)^{l_0} \\
&(dx + ey + fz - dA_x - eA_y - fA_z)^{m_0} \\
&(gx + hy + iz - gA_x - hA_y - iA_z)^{n_0} \\
&= \\
&[a(x - A_x) + b(y - A_y) + c(z - A_z)]^{l_0} \\
&[d(x - A_x) + e(y - A_y) + f(z - A_z)]^{m_0} \\
&[g(x - A_x) + h(y - A_y) + i(z - A_z)]^{n_0}
\end{aligned}
\qquad (0.31)
$$

Thus, the rotated basis functions must be assigned appropriately relocated centers in the new coordinate system.

e) Multinomial Expansions

Expanding a term like $(a+b+c+\cdots)^p$ is easily done by recognizing that the coefficients for various terms in the expansion are given by the following:

$$\left\{\binom{p}{i,j,k\cdots} \equiv \frac{p!}{i!\,j!\,k!\cdots}\right\} \ni i+j+k+\cdots = p \qquad (0.32)$$

where i, j, k, etc. are the powers on a, b, c, etc. and range over all possible values meeting the constraint. In the case of the binomial expansion, $(a+b)^p$, the familiar Pascal's triangle gives a convenient means of quickly coding a routine for the binomial coefficients for moderate p:

1	p=0
1 1	p=1
1 2 1	p=2
1 3 3 1	p=3
1 4 6 4 1	p=4
etc.	etc.

Table 12:
Pascal's triangle for binomial coefficients.

Using a scheme like this in coding is easy and avoids the complications due to evaluating the ratios of large factorials that arise when using the usual combinatorial expressions.

A similar scheme can be devised for trinomial expansions. The visual picture involves a trigonal pyramid with a 1 at the peak (corresponding to p=0) and 1s running down the three edges. Triangular planes slicing through the pyramid parallel to the base contain the trinomial coefficients which are all derived from sums of adjacent numbers in the next higher plane. Therefore, the second layer (corresponding to p=1) is a triangle of three 1s. Some subsequent layers are shown in the following table:

1 2 1 2 2 1	p=2
1 3 3 1 3 6 3 3 3 1	p=3
1 4 6 4 1 4 12 12 4 6 12 6 4 4 1	p=4

Table 13
Pyramid for trinomial coefficients.

The pattern is fairly easy to generate. This also clearly shows how *many* coefficients are needed for a given power, p. (This is generically important for programming since computer memory may now be dynamically allocated to the problem size instead of imposing arbitrary, static upper limits when the program is designed.) The coefficients for a trinomial to the power p reside in a triangular layer with p+1 numbers on an edge. Therefore, the total number of coefficients is just the sum of the first p+1 integers. See equation (7.6).

C. Pipelined Parallelism

If we are content with viewing wave function properties in two dimensional slices, the programmer need not be too concerned with the choice of algorithm since the worst scaling of CPU time goes as $O(n^2)$ where n is the number of grid points along an edge of the square spatial grid of interest. However, in many situations, including the predictions

of STM images it is necessary to have volumetric views of the wave function properties. In this case some part of the calculation will scale as $O(n^3)$ and as we will see, it is necessary generate efficient algorithms in order to obtain these properties in a reasonable amount of time and space (computer memory).

When parallel processing was first introduced to the Goddard group in 1986, we sought ways to increase performance of existing codes with minimal effort. Reproduced here is a trade newsletter article written to describe a method successfully used to reduce the time to solution for one of the group's production codes, the GVB2p5 program (for computing GVB wave functions). The approach is to recognize heterogeneous parallelism in the program flow and exploit it by using multiple processors. A key feature is that this can be done with minimal change to the code.

The impact of these ideas have been limited by a few factors. First, the machinery for implementing shared memory was vendor specific. This results in extra effort for supporting other platforms. Second, newer *ab initio* algorithms have been steadily replacing the use of older codes, primarily on larger computational models where improving time to solution is most important. Third, the mode of production runs in the group has been to let multiple users run jobs simultaneously on a parallel processor. Therefore, unless the machine not saturated (a rare occurrence), the time to solution of a parallel job is not reduced because of time sharing with other users.

Counteracting these trends, today we are seeing the use of shared memory becoming much more transparent. Also, we are seeing some machines, like the Kendall Square Research 64 processor machine, that are being used in more of a batch mode for sets of processors (instead of time shared). Time to solution for individual computations and therefore the ability to use multiple processors for one job is at a greater premium.

Since we are also seeing increased willingness to code parallel algorithms from the ground up, we include our case study as an example of one way to obtain parallelism.

This example also demonstrates what is called heterogeneous parallelism; here, different processors perform distinctly different tasks. This is in contrast to the perhaps more common mode of parallel algorithm development where one looks for ways to operate the same algorithm on different parts of a data set.

Application Pipelining

Terry Coley, California Institute of Technology

The flow of data through many applications can be broken down into a series of sequential "stages" where the output from an earlier stage is fed to the input of subsequent stages. The throughput of many programs exhibiting such data flow can be improved by a simple technique we call "application pipelining". We will show how to put multiple processors to work on a single program and demonstrate the method by describing its application to the "GVB" quantum chemistry code developed by the William A. Goddard III group at Caltech.

The idea of algorithmic pipelining is certainly not new. However, the difficulty of creating automatic methods to divide application programs into pipelined stages has limited its implementation in "production" codes. On the other hand, if one has a basic understanding of the underlying algorithmic structure and access to the source code, we wish to demonstrate here just how simply application pipelining may be accomplished.

There are a few characteristics a program must possess for the technique described here to be useful:

1. The program must perform sequential operations (stages) on a set of data, and these stages should take roughly equal amounts of time for throughput benefits to be seen.
2. Each stage of operation should require at minimum several CPU seconds of processing time to overcome synchronization overhead.
3. The overall set of data to be processed must be divisible into sets such that set M can pass through processing stage N to stage N+1 before set M+1 needs to be processed. The program must use several data sets (or subsets) which may pass independently through the algorithm pipeline.

Our example, the GVB program is a self–consistent field (SCF) electronic wavefunction generating code. The algorithm is an iterative procedure whereby successive refinements are made on a trial wavefunction until a convergence criterion is reached. The time consuming portion of each iteration consists of reading files several hundred megabytes long containing integrals of Gaussian functions and then transofrming these fundamental pieces into Hamiltonians using the current iteration's transformation matrix. The details are not important here: suffice it to say that a significant portion of time is spent reading the integrals in large blocks (called the read stage) followed by a lengthy calculation on each block (called the calculation stage). The results from each block of integrals are independent of others and the raw integrals in memory are discarded before the next block is read in.

This type of program flow is ideally suited to application pipelining. What we would like to do is have one processor reading integrals while another is transforming the previously read block. One could use the complex for such a task, but this would probably be inefficient since the basic number of functional stages we have in mind here is just two: 1) Read and 2) calculate. The complex is often configured larger than two processors. Further inefficiency arises from the fact that the two stages are likely to require somewhat different amounts of CPU time. Once a program is using the Complex, all processors in the Complex are unavailable to do other work even though one is actually idle. Early completion of one stage in the pipeline would result in an empty processor.

Preferable would be a scheme to use a separate detached processor for each pipeline stage and allow stages to communicate their data. In the event that one stage completes before the next is ready, we would like to relinquish the processor to other jobs on the system. The example we show next uses fork(), shared memory regions, and signals to accomplish these goals. Except for the shared memory calls, all code is implemented in FX/FORTRAN for ease of integration with FORTRAN production codes.

The following table shows how a two stage pipeline reduces the time to completion of a pipelined application:

Time ↓	Non-Pipelined Processors →		Pipelined		
	1	R_1	1	R_1	
	2	C_1	2	R_2	C_1
	3	R_2	3	R_3	C_2
	4	C_2	4	R_4	C_3
	5	R_3	5		C_4
	6	C_3			
	7	R_4			
	8	C_4			

Rn represents Read stage n
Cn represents Calculate stage n

Our GVB code uses a large work array for intermediate data storage. At problem initialization time, various indices into this array are computed based on the problem size. One of these indices is used to give the location in the work array of the large input record of integrals. For the pipelined version of the program, we have created a set of utility routines for reading a file into a set of buffers (array locations). The main program simply calls a utility routine instead of its usual READ statement to get a new index to the next

valid input buffer. Thus, since the I/O is done asynchronously with the main program, we call these our asynchronous I/O routines. Use of these routines required about 10 lines of code to be changed in the original program and a small interface routine to be written. The resulting code completes in 25 to 50% the time of the unaltered code.

In summary, usage of the asynchronous I/O routines consists of the following steps:

1. Allocate shared memory regions for a set of control flags and input buffers. Specify the number and size of the input buffers as well as how many records are to be read before the file is rewound. (Recall our application is an iterative procedure.)
2. Fork a child which calls "a_read" This routine will fill input buffers while the parent is calculating
3. To get input for the parent program, call "a_point" which returns a work array index to a filled buffer.

The code in Listing 1 is the interface routine written to set up the program to use the asynchronous I/O routines. In short, the main program must identify a region in its work array that will be used for a set of flags (4096 bytes) and a region for the I/O buffers. Once identified, these information (array locations) is passed to the routine "gvb2_aio" as the arguments "flags" and "buffers". This routine creates system shared memory regions for the flags and buffer areas of memory. Then it forks into two processes. The child will have access to the shared memory regions set up by the parent. The parent records the child process identification and then simply returns to the main program for normal continuation of the algorithm. The child process calls the asynchronous read routine, "a_read", which will use the flags array to direct where to put its input and when to do it.

The program is now primed to operate in pipelined fashion. The following code is extracted from the routine that reads in a chunk of integrals into an array called "ivjk". The variable "aio_on" indicates whether the user has opted to use the asynchronous I/O method. If aio_on is FALSE, then a normal read is done and the offset variable "ioff" is set to zero. The subsequent calculation portions of the program use array "ivjk" always indexed as ivjk(ioff+true_index).

```
if ( aio_on ) then
    call a_point  ( ioff, flags )   ! find a filled data buffer
    ioff = ioff –1                  ! change index to offset
else
    read ( intf ) ivjk              ! do normal FORTRAN I/O
    ioff = 0                        ! zero the offset
endif
```

If aio_on is TRUE, the routine "a_point" simply returns an index into the grand buffer which locates the subbuffer containing the next record of data from the file. The internal bookkeeping is all done using the shared memory page call "flags". These flags are initialized in the call to "a_io_init" and maintain information about which buffers are active for the read and calculate stages of the pipeline.

It is worth noting that internally the "a_read" routine uses the flags to find open buffers for storing data. When no more buffers are available, the child STOPs itself until a signal is sent from the parent. The parent sends a CONTinue signal to the read process each time it frees a buffer (finishes calculations). If the child has been stopped, it now resumes and begins filling the buffer flagged as empty. Likewise, the parent will STOP if it cannot find a filled buffer (as discovered in "a_point"). Since the child sends the parent a CONTinue signal when it fills a buffer, the parent will resume as soon as data becomes available. Note that this allows what would otherwise be idle processors to be released to do other work on the system.

Here are the results for a 10 iteration test case using the GVB2 program to solve for the wavefunction of a C_8H_{14} (cyclooctene) molecule using 100 Gaussian basis functions:

	Elapsed TIme	Corrected Elapsed Time
No Asynchronous I/O	32:16	25:13
Double Buffered	23:37	16:34
% Savings	27%	34%

The corrected row accounts for a 7:03 initialization period for the program where the asynchronous I/O is not used. For larger numbers of iterations commonly used, the true throughput savings approaches the 34% mark. The savings reflects the ratio of I/O time to compute time. This changes with the molecule being solved for. Molecules with more symmetry result in faster computation and a closer match of compute to I/O time. The closer to equal the compute and I/O times are, the closer to the theoretical 50% savings is attained.

In general, for this two stage pipelined program (read, calculate) one would expect no further gain beyond a double buffered setup: read to one buffer while calculating in the other, then swap. In practice, additional marginal throughput gains may be seen from three or four buffers. This happens when the system is more heavily loaded since contention from another user may impede the read process from filling its buffer in

(Applications Pipelining continued on page 12)

(Applications Pipelining continued from page 11)

time during a particular read cycle. Having extra buffers allows the read process to use more idle time on the system to fill a "safety" buffer. If in a particular cycle the compute is ready for more data before the job got enough access to its I/O device to fill a read buffer, it may move on to use the safety buffer while the read process is still contending for I/O time. Of course, in the steady state the I/O speed must keep up with the compute – but additional buffers smooth out local bottlenecks on the system.

The following are the elapsed time results for the same test case as above but using additional buffers. The percent savings figures are corrected as above (by subtracting a 7:03 offset) to show the benefit due to the asynchronous I/O routines:

Number of buffers	Elapsed TIme	Corrected%Savings
1 (no AIO)	32:16	N/A
2	23:37	34.3
3	22:09	40.1
4	20:42	45.9

We have shown how with small changes to the source code, some applications may significantly improve their throughput on Alliant's systems. The asynchronous I/O routines may be obtained from the author by sending your request to "terry@wagfx8.caltech.edu". These routines may be directly applicable to your problem, or they may serve as an initial framework for customizing your application pipelining.

LISTING 1

```
c-----------------------------------------------
c       gvb2_aio.f
c-----------------------------------------------
        subroutine      gvb2_aio        ( iunit, aio_nshared_pages,
        2                               flags, aio_flags_size, buffers,
        2                               aio_buffers_size, nrec_in_file,
        2                               aio_nbuffers )
        implicit        none
        include         'aio.par'
c
c       iunit:
c          FORTRAN unit from which data will be read.
c       aio_nshared_pages:
c          Number of shared memory regions to be allocated for input buffers.
c          The region size (page size) is A_IO_PAGE_SIZE_MAX (2MB on Alliant).
c       flags:
c          Array that will be allocated as shared which stores synchronizing variables
c       aio_flags_size:
c          I*4 dimension of flags().
c       buffers:
c          Beginning of shared input buffer space.
c       aio_buffers_size:
c          Total size of all buffers in I*4 units.
c       nrec_in_file:
c          Number of records in input file to read before rewind.
c       aio_nbuffers;
c          Number of separate buffers to read in.
c
          integer*4     iunit,aio_nshared_pages
          integer*4     aio_flags_size, aio_buffers_size
          inetger*4     nrec_in_file, aio_nbuffers

          integer*4     id, fork, getpid
          integer*4     reg_off, region_base, f_create_shared_region
          integer*4     one_buffer, i, istat

          integer*4     flags (aio_flags_size)
          integer*4     buffers (aio_buffers_size)

c         Iunit for async I/O

          one_buffer                    = aio_buffers_size / aio_nbuffers
          flags (A_IO_SBUF)             = one_buffer * AIO_BUF_TYPE_SIZE
          flags (A_IO_NBUF)             = aio_nbuffers
          flags (A_IO_UNIT)             = iunit
          flags (A_IO_NREC)             = nrec_in_file
          flags (A_IO_PPID)             = getpid ( )
          flags (A_IO_PID)              = 0

          call  a_io_init           ( flags )

c         Get a likely unique shared memory region identifier.

          region_base                   = getpid  ( ) * 100

          1stat = f_create_shared_region ( region_base+1,
          2                                flags, A_IO_FLAGS_BYTES,0 )
          if    ( istat .ne 0 )
          2     stop      'Error creating shared region for aio flags.'
          i     =1
          do    reg_off = 2, aio_nshared_pages+1
                istat = f_create_shared_region
                ( region_base+reg_off, buffers(1), A_IO_PAGE_SIZE_MAX, 0 )
                if      ( istat .ne. 0 )
          2             stop    'Error creating shared region '
                i       = i + A_IO_PAGE_SIZE_MAX / AIO_BUF_TYPE_SIZE
          enddo

          id    = fork ( )
          if    ( id .eq 0 ) then
                call  a_read  ( flags, buffers )
          else
                flags (A_IO_PID) = id
          endif

          return
          end
```