INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
A Comparison of Linear Scaling Replacements for Diagonalization in Electronic Structure Calculations

by

Andrew D. Daniels

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy

APPROVED. THESIS COMMITTEE:

[Signatures and names]

Gustavo Scuseria. Chairman
Professor of Chemistry

John Hutchinson
Associate Professor of Chemistry

Peter Nordlander
Professor of Physics

Houston, Texas

June, 2000
Abstract

A Comparison of Linear Scaling Replacements for Diagonalization in Electronic Structure Calculations

by

Andrew D. Daniels

Even when using parametrized semiempirical methods, quantum chemical calculations on molecules containing more than a few hundred atoms become prohibitively expensive due to $O(N^3)$ time and memory costs where $N$ is the number of atoms. I implemented methods to allow the CPU time cost of semiempirical methods to scale linearly with system size enabling semiempirical calculations on large biological systems such as proteins and nucleic acids. The cost of forming the initial guess density matrix was reduced by replacing the $O(N^3)$ diagonalization of the Hückel Hamiltonian with an approach which uses localized molecular orbitals based on the Lewis dot structure to build the density matrix. The Fock matrix build was reduced from $O(N^2)$ to linear scaling in CPU time using atom-atom distance cutoffs. The diagonalization step was replaced by several linear scaling methods described in the literature: conjugate gradient density matrix search (CGDMS), purification of the density matrix (PDM), pseudodiagonalization (PD), and the
Chebyshev expansion method (CEM). While in my semiempirical implementation all of these methods demonstrated linear scaling, CGDMS, PDM and PD required about the same amount of CPU time for calculations on water clusters and polyglycine chains but CEM was found to be about three times as expensive as the other methods. However, CGDMS stands out among the other methods by having the added property of enhancing self-consistent field (SCF) convergence in cases where diagonalization has convergence difficulties. Finally, to demonstrate the effectiveness of the linear scaling semiempirical method on a realistic system we performed the first-ever semiempirical geometry optimization using PM3 implemented with CGDMS on a 1226 atom kringle 1 of plasminogen.
Acknowledgments

I thank Dr. Gustavo Suseria for his helpful guidance in accomplishing my research goals. I thank John Millam for his help with the development of the CGDMS method, and Kevin Bates for his help with the Chebyshev method.
Table of Contents

Chapter 1: Introduction 1
Chapter 2: Exploiting Matrix Sparsity 7
Chapter 3: Initial Guess Density Matrix 16
Chapter 4: The Electronic Coulomb Problem 19
Chapter 5: Linear Scaling Diagonalization Replacements 24
Chapter 6: Conjugate Gradient Density Matrix Search 30
Chapter 7: Linear Scaling Geometry Optimization of Plasminogen 34
Chapter 8: Convergence Properties of Conjugate Gradient Density Matrix Search 48
Chapter 9: Pseudodiagonalization 62
Chapter 10: Chebyshev Expansion Method 65
Chapter 11: Purification of the Density Matrix 68
Chapter 12: Comparison of Linear Scaling Methods 72
Chapter 13: Conclusions 89
References 91
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Matrix sparsity for polyglycine chains.</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Matrix sparsity for water clusters.</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Energy fluctuations induced by Fock matrix cutoffs.</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4</td>
<td>RMS gradient during plasminogen optimization.</td>
<td>42</td>
</tr>
<tr>
<td>Figure 5</td>
<td>RMS gradient near end of plasminogen optimization.</td>
<td>43</td>
</tr>
<tr>
<td>Figure 6</td>
<td>RMS displacement during plasminogen optimization.</td>
<td>44</td>
</tr>
<tr>
<td>Figure 7</td>
<td>RMS displacement near end of plasminogen optimization.</td>
<td>45</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Heat of formation during plasminogen optimization.</td>
<td>46</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Heat of formation near end of plasminogen optimization.</td>
<td>47</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Comparison of the CPU time requirements of linear scaling diagonalization replacements for polyglycine chains.</td>
<td>82</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Comparison of the CPU time requirements of linear scaling diagonalization replacements for water clusters.</td>
<td>83</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Comparison of the CPU time requirements of linear scaling diagonalization replacements for RNA molecules.</td>
<td>84</td>
</tr>
<tr>
<td>Figure 13</td>
<td>The number of SCF iterations required for SCF convergence when using various linear scaling diagonalization replacements for polyglycine chain calculations.</td>
<td>85</td>
</tr>
<tr>
<td>Figure 14</td>
<td>The number of SCF iterations required for SCF convergence when using various linear scaling diagonalization replacements for water cluster calculations.</td>
<td>86</td>
</tr>
</tbody>
</table>
Figure 15. The number of SCF iterations required for SCF convergence when using various linear scaling diagonalization replacements for RNA molecule calculations.

Figure 16. Memory requirements for the linear scaling diagonalization replacements in polyglycine chain calculations.
List of Tables

Table 1. Geometric parameters for CrC, Cr$_2$, UO$_2$(OH)$_1$ and UF$_1$. 59

Table 2. Effect of an added VShift on SCF convergence for CGDMS and diagonalization. 60

Table 3. Comparison of the SCF convergence properties of CGDMS and diagonalization with various VShifts. 61

Table 4. Number of purifications cycles required as a function of the CPO in CEM. 71

Table 5. CPU cost of single point energy calculations on several large molecules. 81
Chapter 1

Introduction

My research involved adapting, implementing and comparing linear scaling techniques for replacing diagonalization in semiempirical quantum chemical calculations[1, 2, 3, 4, 5]. Quantum chemical calculations describe chemical properties of molecular systems such as reaction enthalpies, optimum geometries, charge distributions and bonding[6]. One major drawback of conventional quantum chemical calculations is that they are only practical for systems with small numbers of atoms[1]. Semiempirical methods were developed to deal with this problem[7, 8]. These methods are a result of making several additional approximations to the Hartree Fock method such as using a valence-only basis set and neglecting integrals involving diatomic differential overlap. The remaining integrals are then fitted to experimental ionization and atomization energies to increase the accuracy and efficiency of the calculations and to include electron correlation effects[7, 8]. In spite of all of the approximations made, semiempirical methods such as AM1[7] or PM3[8] gave more accurate results than Hartree Fock at a mid-sized basis set of 6-31+g(d,p) in calculating thermodynamic data such as atomization energies, ionization potentials, electron affinities, and proton affinities for a set of 125 calculations[6]. The molecules for these calculations where chosen because they had very accurate experimental results and contained a range of elements and electronic states[6]. One
explanation for the poor performance of the Hartree Fock method is that it does not include correlation energy, whereas AM1 and PM3 are parametrized to include it implicitly[7, 8].

However, even with all of these approximations, the computer time required for semiempirical calculations scales as $O(N^3)$ with system size, where $N$ is the number of atoms[1]. This limits the size of the chemical systems that can be practically calculated to a few hundred atoms, thus precluding calculations on many chemically and biologically interesting systems. Most of these large systems are currently dealt with using molecular mechanics methods, which use Newtonian physics to model the molecules in a completely empirically parametrized fashion[9, 10]. However, many chemical properties such as charge distribution and bond breaking cannot be determined accurately using this kind of modeling,[9, 10] To remedy this, researchers are developing methods such as ONIOM[11] which facilitate calculations on large molecules by partitioning the molecule and using high accuracy methods for small regions of interest, while handling the rest of the molecule using a cheaper computational method[11]. However, these methods are not black-box. It is necessary to define portions of the molecule whose chemical properties are important. It is not always clear as to which portions of the molecules must be treated quantum chemically and which are not as important. Furthermore, in my experience with using the ONIOM code I have found that it has self-consistent field (SCF) convergence problems. These problems are partly caused by the fact that the boundaries of the various regions of the molecule which are treated with different levels of theory must be terminated with hydrogens or other groups which may not lead
to a physical description of the system. On the other hand, if quantum chemical methods could be made more efficient, the entire system could be treated quantum chemically making the calculation black box and thus avoiding many of these problems.

In order to accomplish the goal of creating linear scaling semiempirical methods, one must take a detailed look at the inner workings of the calculation [12]. There are three major matrices involved in the self-consistent semiempirical calculation which are all in the minimum valence atomic orbital basis. The core Hamiltonian matrix $(H)$ contains the one-electron integrals of the chemical system to be calculated. This matrix describes the kinetic energy of the system and the electron nuclear attraction interactions. Since these properties depend only on the positions and charges of the nuclei, $H$ is independent of the electron density and is formed only once at the beginning of the calculation. The Fock matrix $(F)$ consists of $H$ plus the two electron integrals which depend on the electron density of the system. This matrix describes the Coulombic interactions of electrons in an average electronic field. It also contains exchange interactions which arise from the antisymmetry of the wavefunction.

The density matrix $(P)$ is the fundamental variable in SCF calculations. It is used to describe the average electron density when forming $F(P)$, and is used in projecting out the virtual subspace of $F$ when calculating the energy. This matrix is defined by the Fermi–Dirac distribution:

$$ P = \frac{1}{1 + e^{(F - \mu)}} \quad (1.1) $$
where $\mu$ is the Fermi level and $\beta$ is the fictitious inverse temperature. In semiempirical calculations the temperature is taken to be zero so $\beta$ is infinite and equation 1.1 is the step function with the occupied subspace having a functional value of 1 and unoccupied subspace having a functional value of 0. In conventional semiempirical calculations, the density matrix is formed from the canonical orbitals, $\psi$, which are the orthonormal eigenvectors which diagonalize the Hermitian matrix, $F$:

$$F \psi = \psi E.$$  \hspace{1cm} (1.2)

where $E$ is a diagonal matrix containing the eigenvalues of $F$. $P$ is obtained using an outer product of the $N_e$ eigenvectors (where $N_e$ is the number of electrons) corresponding to the lowest energy eigenvalues of $F$.

$$P_{ij} = \sum_{k=1}^{N_e} \psi_{ik} \psi_{jk}$$ \hspace{1cm} (1.3)

Since $F(P)$ and $P$ is formed from diagonalizing $F$, the semiempirical calculation is nonlinear and is most commonly solved iteratively via a SCF procedure. The SCF procedure involves four steps: $F$ is built with an initial guess $P$, $F$ is diagonalized, $P$ is formed, and the energy is calculated and checked for self-consistency. This process is repeated using the $P$ from the previous SCF iteration to form each new $F$ until the the procedure is self-consistent. In other words, the calculation is terminated when a norm of the change in the density matrix from one iteration to the next is below a certain threshold. The converged density matrix can be used to obtain physical properties of the chemical system such as the energy, the electron distribution, dipole and higher moments, bond orders, and derivatives of the energy with respect to nuclear displacement. The goal of forming a linear
scaling semiempirical method can be accomplished by reducing each of the steps within the SCF procedure to linear scaling with system size.

Most of the linear scaling work in the literature is focused on two problems. The first problem deals with reducing the scaling of the electronic Coulomb and exchange–correlation calculations within density functional theory methods to linear with system size[13, 14, 15, 16, 17, 18, 19, 20], while the other deals with reducing the time scaling of the diagonalization step in the tight-binding method to linear with system size[2, 21, 22, 23, 24, 25, 26]. This tight-binding problem is much simplified by the fact that it is not a self-consistent method, and its matrices contain very short-ranged interactions where the interaction range is known before the calculations is performed[27]. Thus, there may be significant differences in the implementation and performance of the linear scaling methods between the tight-binding case and the more complicated self-consistent case. Besides work on pseudodiagonalization[28] and work on the divide and conquer method[29, 30, 31, 32], very little has been done on reducing the scaling of SCF semiempirical calculations to linear with system size.

This work describes my linear scaling implementation of the semiempirical methods and focuses on comparing four linear scaling diagonalization replacements. Chapter 2 illustrates how linear scaling is made possible through exploiting the locality of interactions. The principles in this chapter are used to obtain linear scaling in the diagonalization step as well as in calculating the energy of the system. Chapter 3 discusses how the initial guess \( P \) can be formed in a fast, efficient manner based on a localized molecular orbital description of the system. Chapter 4
describes my implementation of a method to form $F$ and $H$ in linear time. Chapter 5 introduces the conjugate gradient density search method for replacing diagonalization and Chapters 6 and 7 give applications of this method. Chapters 8–10 introduce three other linear scaling diagonalization replacements while Chapter 11 gives a comparison of their performance and advantages and disadvantages. Finally, Chapter 12 provides some concluding remarks.
Chapter 2

Exploiting Matrix Sparsity

The main purpose of my research was to implement and compare linear scaling replacements for diagonalization in semiempirical calculations. Any linear scaling computational technique must contain a linear number of floating point operations (flops) in each step of the calculation. This condition requires that the number of significant matrix elements involved in the calculation must scale no worse than linearly with system size. This concept is obvious because any operation which manipulates more than a linear number of matrix elements would necessitate more than a linear number of flops. This chapter will discuss these concepts in more detail and show how they relate to linear scaling semiempirical quantum chemical calculations.

The fact that quantum chemical methods use $N \times N$ matrices suggests that, when coded in the obvious fashion, the calculation will have at least $O(N^2)$ memory storage and CPU time costs. However, linear scaling in both time and memory for many linear algebra operations can be achieved through utilizing sparse matrix techniques if the number of significant matrix elements increases linearly with the size of the system[33].

The question then becomes which matrix elements are significant? In practice, very few of the matrix elements in matrices involved in semiempirical calculations
are exactly zero. However, many matrix elements are so small that they do not contribute to the desired accuracy of the calculation. Most implementations of linear scaling replacements for diagonalization in the literature use atomic distances to determine which matrix elements are significant and which ones can be safely neglected[2, 21, 22, 23, 24, 25, 26]. When a certain matrix element represents interactions between atoms which are further apart than a certain atom atom distance (the cutoff distance) it is considered to be zero. This method is particularly well suited in tight-binding calculations since by definition their Hamiltonians contain only very short-ranged interactions. For instance, the tight-binding carbon-carbon potential is zero beyond 2.6 Å[27]. The density matrix is more delocalized than the tight-binding Hamiltonian so its cutoff distance must be around 4 Å to obtain accurate results[27]. However, interactions in self-consistent semiempirical calculations such as AM1 or PM3 have a much longer range. Thus, to obtain reasonable accuracy (µHartree to mHartree) the cutoff distance must be set to about 15 to 20 Å[1]. This is very inefficient because even though some of the matrix elements at these distances are significant, many of them are negligible. Thus, a more suitable approach for semiempirical methods consists of throwing away matrix elements whose absolute values are below a certain neglect threshold[33, 34]. This technique is independent of the particular matrix form and thus is black box.

The speed and accuracy of the calculation depends on the size of the neglect threshold or the length of the cutoff distance. The larger the neglect threshold or shorter the cutoff distance, the more matrix elements are discarded and the faster the calculation. However, the more matrix elements that are discarded, the further
the computational results will be from the full matrix result. Thus, there is a trade off between the accuracy and speed of the calculation. I have found that neglect thresholds between $5 \times 10^{-5}$ a.u. and $1 \times 10^{-7}$ a.u. and a distance cutoff of about 15 A are good compromises between speed and accuracy in semiempirical calculations[1].

Matrix sparsity can be exploited by storing only the nonzero elements of the matrix which I define to be those elements whose absolute values are above a certain neglect threshold or those within a certain physical cutoff distance. Because not all of the matrix elements are stored, the matrices can no longer be indexed as full matrices are, and it is thus necessary to determine the row and column positions of each matrix element. These positions must be easily accessible with as little searching as possible so that the matrix algebra such as matrix addition and multiplication can be done efficiently in linear time. The most straightforward way of doing this is to store the matrices in bands. Given a matrix $A$, with matrix elements $A_{ij}$, the center band is an array containing the diagonal elements of the matrix, $A_{ii}$, the $(N - 1)$th band is an array containing the elements $A_{i,i-1}$, and so forth. This matrix storage technique provides a fast and efficient method to access matrix elements without much overhead from indexing. However, the matrices involved in quantum chemical calculations are not strictly banded. This means that even though most of the larger matrix elements are near the diagonal of the matrix, there are still quite a few significant matrix elements scattered throughout the entire matrix. Thus, another method must be used to store the matrix.
The method which I used to store sparse matrices is a modification of the method suggested by Pissanetsky[33]. In this method, each matrix is stored as two integer indexing arrays, \( IA \) and \( JA \), along with a real array, \( AN \), containing the nonzero matrix elements. With these three arrays, along with \( N \) defined as the dimension of the matrix and \( NZ \) as the number of nonzero elements of the matrix, all of the information about the matrix elements and their indices can be determined. \( IA \) is dimensioned \( N + 1 \) and contains an index into \( JA \) and \( AN \) for the position of the first element of each row in the matrix. \( JA \) is of length \( NZ \) and contains the column positions of the matrix elements stored in \( AN \). Using this storage method it is possible to access matrix elements with very little searching during matrix operations.

Storing matrices in the above method not only reduces the amount of computer memory required, but it also reduces the time cost of matrix operations such as scalar multiplication, matrix-matrix addition, and matrix multiplication. Let us assume that the number of nonzero elements (\( NZ \)) in matrices \( A \) and \( B \) increases linearly with the dimension (\( N \)) of the matrix. By utilizing sparse matrix storage, the CPU time cost of the above matrix operations on \( A \) and \( B \) can be made to scale linearly with system size. Take, for instance, the matrix product, \( C = A \times B \). Since \( NZ \) increases linearly with \( N \), the average number of nonzero elements in each row of \( A \) and \( B \) is constant with increasing \( N \) (\( NRow = NZ/N \)). A matrix multiplication can be thought of as a series of \( N \) matrix-vector multiplications. To show that the time cost of matrix multiplication increases linearly with increasing \( N \), we need only demonstrate that the time cost of each of the matrix-vector prod-
ucts is constant with increasing $N$. This matrix vector cost is constant because each vector contains a constant number of elements, $N_{Row}$, which need only be multiplied with those matrix rows having nonzero elements in positions corresponding to the nonzero vector elements. Since the number of nonzero elements in each matrix row is also constant with increasing $N$, the cost of the matrix vector product is $O(N_{Row} \times N_{Row})$ which is indeed a constant. Therefore, the CPU time cost of matrix multiplication is $O(N \times N_{Row} \times N_{Row})$ and thus scales linearly with increasing $N$. It is also useful to notice that for a given $N$, the cost of matrix multiplication increases quadratically with the number of matrix elements in each row.

There are several issues involved when matrix operations are performed using sparse matrices instead of full ones. These issues include accuracy, overhead and matrix fill-in. As mentioned before, the accuracy of sparse matrix calculations depends on the criterion for throwing away matrix elements. Additionally, sparse matrix techniques are less efficient than linear algebra performed with full matrices because of the cost of indexing. This overhead is incurred mostly in performing the "symbolic operation" which is used to determine the positions of the nonzero elements in the result of the matrix operation as well as to allocate memory for the new matrix. The symbolic operation is often as expensive as the numerical part itself. The third issue is matrix fill-in. For example, in $C = A \times B$, $C$ often has a considerable number of nonzero elements in positions where both $A$ and $B$ have zero elements. However, a lot of these "new" elements are below the neglect
threshold and thus can be discarded. This causes matrix growth and is kept in check by thresholding the matrices after each matrix multiplication.

A significant advantage can be obtained if the form of the matrix product is known beforehand as it is in tight-binding methods when using cutoff distances to determine the sparsity of the matrix. If the form of the resulting matrix is known beforehand, the cost of the matrix operation can be greatly reduced by skipping the symbolic matrix manipulations, as well as by avoiding unnecessary matrix fill in and allowing more efficient memory allocation. Using a fixed form for matrix multiplication normally increases the speed of the operation by more than a factor of 2. In later chapters I will discuss more about how a fixed matrix form can be used in semiempirical methods to create efficient linear scaling diagonalization replacements.

I will now look at the sparsity of the particular matrices in the semiempirical calculation. I first consider how to introduce sparsity into $F$. The $F_{\mu \lambda}$ matrix elements (where $\mu$ and $\lambda$ are on different atoms) contain the two-center one-electron core resonance integrals and two-center two-electron repulsion integrals which both die off exponentially with increasing distances between atoms[1]. Thus, $F$ is sparse and its number of matrix elements scales linearly with system size. Figures 1 and 2 demonstrate this linear scaling by presenting the number of surviving elements of $F$ and $P$ for a number of neglect thresholds and system sizes for polyglycine chains[35] and water clusters[36], respectively. Note that the polyglycines are not in the $\alpha$-helix conformation, but are extended linear chains. In practice, while calculating $F$ it is desirable to know the form of the matrix beforehand. Thus,
a cutoff distance is used to determine whether or not to calculate a particular element in \( F \). I found that a 10-15 \( \AA \) atom atom cutoff applied to these Fock matrix off-diagonal elements in insulating systems only causes an \( \mu \)Hartree error in the energy as compared to an infinite cutoff distance[1]. Additional elements of \( F \) are eliminated later by discarding all elements below a certain neglect threshold.

However, the eigenvectors formed from diagonalizing \( F \) are in principle non-sparse even though \( F \) may be sparse. This is due to the requirement that the eigenvectors must be orthonormal with one another. Thus, to form a linear scaling semiempirical method, matrix diagonalization must not be used and the canonical orbitals must not be formed.

\( P \), on the other hand, is local in real space. In fact, for insulators, \( P \) decays exponentially with system size, while its decay is a power law for metals[21]. Thus, \( P \) should be sparse for insulating systems and its number of elements should increase linearly with system size. Figures 1 and 2 demonstrate that the number of significant matrix elements in \( P \) does indeed increase linearly with the system size for polyglycine chain and water cluster calculations. One thing worth noting is that the matrices obtained in polyglycine calculations are much more sparse than those produced in water cluster calculations at the same neglect thresholds. This occurs because polyglycine chains are linear systems which are physically much more spread out than water clusters. In conclusion, it is possible to achieve linear scaling in semiempirical methods if matrix diagonalization is replaced with techniques which use linear scaling matrix operations form \( P \) directly without forming the eigenvectors of \( F \).
Figure 1  Number of surviving matrix elements as a function of the neglect threshold (a.u.) for polyglycine chains using CGDMS AM1.[1]
**Figure 2** Number of surviving matrix elements as a function of the neglect threshold (a.u.) for water clusters using CGDMS AML.[1]
Chapter 3

Initial Guess Density Matrix

The first non-linear scaling step in a self-consistent field (SCF) quantum chemical calculation is the formation of the initial guess density matrix \( P_0 \). Gaussian forms \( P_0 \) for semiempirical calculations by a \( O(N^4) \) scaling diagonalization of the Hückel Hamiltonian. This chapter discusses methods for computing \( P_0 \) efficiently removing this \( O(N^4) \) bottleneck.

Before discussing how to form alternative \( P_0 \)'s, it is useful to mention the properties of a good density matrix guess. A physical density matrix satisfies the following conditions:

- Idempotency. \( P^2 = P \)
- Normalization. \( Tr(P) = N_e \), where \( N_e \) = Number of electrons
- Commutation with the Fock matrix \( (F) \). \( FP = PF \).

Also, when the density matrix satisfies the Aufbau principle occupations it has the added property:

- Minimum energy \( (Tr(PF) = \text{min}) \).

The goal is to inexpensively form a \( P_0 \) which would come sufficiently close to meeting these criteria so that the SCF cycles will rapidly converge to the correct solution. The density matrix formed by diagonalizing the Hückel Hamiltonian meets the first
two criteria exactly. However, this simple Hamiltonian is much different from the converged $F$ so it gives a poor description of the system's interactions and does not meet the last two criteria. Since the density matrix contains information about the electron density of the systems, this property is less than optimally described in the Hückel $P$. However, this guess is sufficient to allow the SCF to converge in most of the calculations in this work.

Of all of the above criteria for density matrices, normalization is the easiest to satisfy. Thus, the simplest possible replacement for $P_n$ is a diagonal matrix which completely neglects the idempotency condition, only satisfying the normalization condition: $Tr(P_n) = N_e$, where $N_e$ is the number of electrons. Essentially, in an uncharged system, this density matrix is formed by allocating enough electrons to each of the atoms' orbitals so that a neutral atomic charge is maintained on each of the atoms without any regard to their chemical environment. I used this diagonal $P_n$ as a guess for semiempirical AM1 calculations using the conjugate gradient density matrix search technique (CGDMS) for replacing diagonalization. This initial guess density matrix was sufficient to converge the SCF procedure for small molecules and very sparse systems such as linear polyglycine chains. However, with large or compact systems such as large graphitic sheets and diamond chunks, purification transformations failed to make the initial diagonal density matrix idempotent[1]. This occurs because CGDMS is not explicitly constrained to be idempotent.

To circumvent this problem, John Millam implemented a method to form the initial guess density matrix[1] similar to a procedure proposed by Stewart[28]. The molecule's Lewis dot structure is used as a model to form localized molecular or-
bitals from linear combinations of atomic orbitals on nearest neighbor atoms forming bonding and antibonding orbitals. The bonding and lone pair orbitals then are used to construct an initial guess density matrix. Since the molecular orbitals are orthonormal, this density matrix satisfies both normalization and idempotency. This density matrix proved to be sufficient for finding the correct energy minimum in most of the molecules tested in this work. John Millam’s implementation of this method for building the density matrix scales quadratically with system size but the timing is negligible compared to the largest systems studied because of its very small prefactor.
Chapter 4

The Electronic Coulomb Problem

The cost of forming the core Hamiltonian ($H$) and Fock matrices ($F$) is $\mathcal{O}(N^2)$ in semiempirical methods such as AM1 and PM3[7, 8]. This cost is due to describing the interactions of one atom with all of the other atoms in the system and can be reduced to $\mathcal{O}(N)$ in a number of ways[1, 13, 14, 29]. The simplest method for calculating $F$ and $H$ in linear time is to introduce an atom–other atom cutoff criterion which defines a physical separation beyond which all interactions are considered to be zero[1, 29]. A more elegant method is the fast multiple method (FMM) which works by expanding the Coulombic interactions in a series of higher hierarchical multiple expansions and interacting these multiples with one another using a tree code algorithm[13, 14]. This chapter deals with the method I used to reduce the time and memory scaling of the electronic Coulomb problem to linear with system size for semiempirical methods.

The first problem in computing the electronic Coulomb problem with linear memory requirements is the method by which the two–center two–electron integrals are stored. In GAUSSIAN's[37] original version of AM1 with diagonalization, the two–center two–electron integrals were stored in memory throughout the calculation which requires about $50 \cdot N^2$ words of memory where $N$ is the number of atoms. This storage becomes prohibitively expensive in large systems. I eliminated this
expensive storage by recalculating the two-center two-electron integrals every SCF cycle, contracting them with the density matrix, and storing them in the Fock matrix[38]. The time penalty that this integral recalculation introduces is a minor contribution to the entire calculation.

The second issue in reducing the electronic Coulomb problem to linear scaling is in calculating the integrals. As mentioned in Chapter 2, sparsity is introduced in \( F \) by applying a \( F_{\mu\lambda} \) atom-atom cutoff criterion (\( Cutoff f_{\mu\lambda} \)), where \( \mu \) and \( \lambda \) are basis functions on different atoms[1, 3]. However, the Fock matrix formation step still scales quadratically because of an \( \mathcal{O}(N^2) \) loop in calculating the two-center two-electron integrals which contribute to the \( F_{\mu\nu} \) elements, where \( \mu \) and \( \nu \) are basis functions on the same atom. By introducing a cutoff criterion (\( Cutoff f_{\mu\nu} \)) into this loop the scaling of the Fock matrix build becomes linear with system size. However, the two-center contributions to the \( F_{\mu\nu} \) matrix elements die off quite slowly with increasing distance between atoms. Calculating the full two-center Coulomb contribution to the \( F_{\mu\nu} \) matrix elements causes the Fock matrix formation step to scale quadratically. Dixon and Merz (DM)[29] tested the possibility of introducing \( Cutoff f_{\mu\nu} \) in this \( \mathcal{O}(N^2) \) loop beyond which all interactions are considered to be zero. They conducted a study on 25-glycine using cutoff distances ranging from 20 Å to 100 Å. Oscillations in energy were seen on the order of \( \pm 0.04 \) Hartrees and \( \pm 0.02 \) Hartrees for 30 Å and 50 Å cutoff distances respectively. DM concluded that use of the zwitterion form of 25-glycine was the reason for the oscillations. DM also hypothesized that since the oscillations in energy appear to be repeated in regular intervals approximately equal to the length of a C-terminal glycine residue.
important interactions were alternately included and excluded with the positive N-terminal residual and the negative C-terminal residue, respectively.

I performed calculations on the non-zwitterion form of 25-glycine using AM1 which revealed oscillations in energy that were about three times smaller than what DM found. I attribute this reduction in energy oscillations to my use of the non-zwitterion form of polyglycine. The fact that oscillations do occur indicates that they are also prevalent in neutral molecules. Oscillations in energy increase significantly with increasing system size: in 100-glycine calculations they are about 7 times larger than those found in 25-glycine, being on the order of ±0.1 Hartrees at a 30 Å cutoff distance. The large error that is introduced by $C_{\text{soft}}$ as the system size increases clearly precludes its use for building the Fock matrix in linear time.

To reduce this error, I introduced a “soft” $F_{\mu\nu}$ cutoff criterion ($SoftC_{\text{soft}}$) in which the full $F_{\mu\nu}$ Coulomb contribution to the Fock matrix is computed up to a certain atom-atom distance, and then is gradually scaled to zero as a function of the distance between atoms for the length of a “soft tail”. Several functions were tested for use as this soft tail including linear, parabolic, exponential, Gaussian, and cubic polynomial functions. The best energy agreement with calculations performed using no $SoftC_{\text{soft}}$ was obtained using a 10 Å cubic polynomial tail. In this work $SoftC_{\text{soft}}$ is defined by an atom-atom cutoff distance that is terminated with a 10 Å cubic polynomial soft tail, where the length of $SoftC_{\text{soft}}$ is measured from the atom in question to the end of the soft tail. Figure 3 shows results from AM1 calculations on 25-glycine as a function of $SoftC_{\text{soft}}$. An order of
magnitude improvement in the energy is gained over calculations using $C_{\text{utoff}} f_{\mu\nu}$. $SoftCutoff f_{\mu\nu}$ has the effect of averaging the contributions of the Fock matrix elements within the cutoff region. $SoftCutoff f_{\mu\nu}$ will be used in this work to demonstrate the effect of a linear scaling Fock matrix build.
**Figure 3** The difference in single point energies (Hartrees) of 25-glycine between diagonalization and CGDMS AM1 with various values for $SoftCutoff_{\mu\nu}$ and $Cutoff_{\mu\nu}$ (Å).[1]
Chapter 5

Linear Scaling Diagonalization Replacements

As mentioned in Chapter 1, one of the steps in a self-consistent quantum chemical calculation is diagonalizing the Fock matrix \( F \). From the eigenvectors of \( F \) the density matrix \( P \) is formed which contains information about the electron density of the system and is the fundamental variable of the self-consistent procedure. The main focus of my research was to reduce the time scaling of the diagonalization step in quantum chemical calculations to linear with system size. Diagonalization is inherently a procedure whose time scaling is \( \mathcal{O}(N^3) \) (where \( N \) is the number of atoms) since it involves forming orthonormal eigenvectors which are generally non-sparse even though the Fock matrix \( F \) may be sparse. It is possible to calculate a single eigenvector in \( \mathcal{O}(N) \) time using Lanczos techniques\[39, 40, 41\]. However, when forming more than one eigenvector, these methods scale as \( \mathcal{O}(N \times n^2) \), where \( n \) is the number of eigenvectors obtained. Since for quantum chemical calculations, \( n \) is the number of electrons in the system \( (N_e) \), and \( N_e \) increases with the size of the system, the Lanczos method is still a \( \mathcal{O}(N^3) \) method. This chapter introduces methods in the literature for forming \( P \) in linear time by finding ways around forming the eigenvectors of \( F \).
Before doing so, I will give a review of the definition and properties of \( P \) which were introduced in Chapters 1 and 3. \( P \) is defined by

\[
P = \frac{1}{1 + e^\beta(E - \mu)}
\]  

(5.1)

where \( \mu \) is the Fermi level and \( \beta \) is the fictitious inverse temperature and is conventionally formed by first diagonalizing \( F \)

\[
F \psi = \epsilon \psi.
\]  

(5.2)

and then forming \( P \) from an outer product of the \( N_e \) eigenvectors corresponding to the lowest energy:

\[
P_{ij} = \sum_{k=1}^{N_e} \psi_{ik} \psi_{jk}^\dagger
\]  

(5.3)

\( P \) has the following properties:

- Idempotency. \( P^2 = P \)
- Normalization. \( \text{Tr}(P) = N_e \), where \( N_e \) = Number of electrons
- Commutation with the Fock matrix \( (F) \). \( FP = PF \).

Also, when the density matrix satisfies the Aufbau principle occupations its trace with \( F \) is a minimum:

- \( \text{Tr}(PF) = \text{Minimum} \).

My goal is to find linear scaling methods for forming a matrix \( P \), which meets all of these criteria.

There are several ways of accomplishing this goal. One method proposed by Dixon and Mertz[29] is to divide up the chemical system into a series of overlapping
subsystems. These subsystems usually are chosen as logical subunits of the chemical system such as individual amino acids in a protein or nucleic acids in DNA. These subunits are then treated as separate systems and diagonalized independently of one another. Afterwards, they are added together taking into consideration the overlapping regions. This method will not be considered in this thesis because its error with respect to the fully diagonalized system is not predictable. Even though this error goes to zero as the subunit size approaches the size of the chemical system, it is not a predictable function of the subunit size.

Another method for removing the $O(N^3)$ diagonalization step is by calculating localized molecular orbitals instead of the canonical orbitals. This method was first developed by Stewart, Császár, and Pulay[42] and was later adapted as a linear scaling method for semiempirical methods by Stewart[28]. The density matrix contains only two eigenvalues, 0 and 1, where the eigenvalue of 1 corresponds to the occupied orbitals, and the eigenvalues of 0 to unoccupied orbitals. Since the eigenvalues are degenerate, any linear combination of the canonical orbitals is also a solution. Thus, it is possible to construct a localized set of molecular orbitals using unitary transformations of the canonical orbitals[43]. Pseudodiagonalization (PD) is a method which implicitly forms localized orbitals by performing a series of Jacobi rotations between the occupied and virtual subsets of orbitals from the last SCF iteration[3, 28, 42]. Because only a linear number of rotations are needed each self-consistent field (SCF) cycle, and because the orbitals are localized, PD scales linearly with system size.
The next three linear scaling methods differ in that they involve calculating the density matrix directly without going through the molecular orbitals. From this $P$ it is possible to compute all of the properties of the system. The advantage of this is that the density matrix is sparse, as was discussed in the chapter on matrix sparsity. Thus, one large family of linear scaling techniques exploits the sparsity of $P$ by finding alternative methods for forming it without explicitly forming the molecular orbitals.

The first of these density matrix methods is the conjugate gradient density matrix search (CGDMS). Li, Nunes and Vanderbilt first proposed CGDMS for the non-self-consistent tight-binding method[21, 22]. Later I adapted it to work for self-consistent semiempirical methods[1], while Millam adapted it for density functional theory methods[34]. CGDMS works by minimizing a functional of $P$ using the method of conjugate gradients. This functional has the property that at its minimum it meets all of the criteria of a density matrix: idempotency, normalization, commutation with $F$, and a minimum energy. By exploiting the sparsity of the matrices involved, CGDMS scales linearly with the system size.

The second method, Chebyshev expansion method (CEM), which Goedecker and Colombo developed for the tight binding method and I later adapted for SCF semiempirical methods forms $P$ as an expansion of a function of $F$[2, 3, 23, 24]. As mentioned above, the density matrix can be defined by Equation 5.1. This equation is expanded in a Chebyshev polynomial of $F$ giving the density matrix. CEM scales linearly with system size when sparse matrix techniques are used.
Palsen and Manolopoulos[25] developed a third method which differs from the other two in that it meets the properties of the converged density matrix in a two step fashion. First, a guess for the density matrix is formed which commutes with the Hamiltonian and has its largest eigenvalues corresponding to occupied states and its smallest eigenvalues corresponding to unoccupied ones. This matrix is then made idempotent by a series of WeWeenyp purification iterations which do not destroy the commutation properties of the density matrix. The final matrix thus meets all of the requirements of the density matrix: commutation with $F$, idempotency, and normalization.

Another promising method for replacing diagonalization is the sign matrix method[44, 45]. The sign matrix is similar to the density matrix in that it is formed from an outer product of the eigenvectors of $F$. The difference between the sign matrix and the density matrix lies in the eigenvalues of the occupied and virtual subspaces. Whereas the density matrix has eigenvalues of $-1$ for the occupied subspace and $0$ for the virtual, the sign matrix has eigenvalues of $-1$ for the occupied subspace and $+1$ for the virtual. The sign matrix method is similar to PDM in the way the matrix is formed, but it can be done in such a way as to take greater advantage of matrix sparsity[44]. However, because of the similarities between this method and PDM, in this work I will not deal with the sign matrix method.

The following chapters will describe several of these methods in more detail. Although the method comparisons were all done in semiempirical methods which have orthogonal minimal valence bases, these methods can be trivially generalized
to a non-orthogonal basis by transforming the Fock and density matrices to the orthogonal basis using the Cholesky and inverse Cholesky matrices before performing the various linear scaling methods[34].
Chapter 6

Conjugate Gradient Density Matrix Search

The first linear scaling diagonalization replacement method I worked with is a density matrix method called conjugate gradient density matrix search (CGDMS). My implementation of CGDMS differs from that of Li, Nunes and Vanderbilt[21] who developed it for the tight-binding method in that it is adapted for the self-consistent case. However, it is similar to that of Millam and Scuseria[34] who adapted CGDMS for use with density functional theory except that the semiempirical methods which I dealt with have a basis which is assumed to be orthogonal so there is no need to deal with transformations to and from the orthonormal basis.

The idea behind CGDMS is to do a functional minimization instead of a matrix diagonalization to form the density matrix, $P$. To implement this idea, one must first find a functional of some guess density matrix ($P_a$) which, when minimized with respect to $P_a$, produces a $P$ which satisfies all of the requirements of a density matrix. As mentioned in previous chapters, the density matrix produced from an Aufbau principle ordering of eigenvectors has 4 properties:
• Idempotency, $P^2 = P$:

• Normalization, $Tr(P) = N_e$, where $N_e$ is the number of electrons:

• Commutation with the Fock matrix ($F$). $FP = PF$; and

• Minimum energy ($Tr(PF) = \text{Minimum}$).

The following functional, $\Omega(P)$, with its gradient meets all of these requirements at its minimum:

$$\Omega(P) = tr(\hat{P}F) + \mu(tr(P) - N_e)$$  \hspace{1cm} (6.1)

$$\hat{P} = 3P^2 - 2P^4$$  \hspace{1cm} (6.2)

$$\frac{\delta \Omega}{\delta \hat{P}} = 3(FP - PF) - 2(FP^2 + PF^2 + P^2F) + \mu I$$  \hspace{1cm} (6.3)

where $\hat{P}$ is a refined density matrix obtained through the McWeeny purification transformation[46] and $\mu$ is chosen such that the gradient is traceless. At a minimum, normalization is trivially satisfied because of the choice of $\mu$. Commutation with $F$ is satisfied because when $\frac{\delta \Omega}{\delta \hat{P}}$ is zero, $F$ and $P$ commute. A minimum trace of $P$ with $F$ is satisfied because the functional contains $Tr(PF)$ (When $P$ is idempotent, $\hat{P} = P$). The idempotency condition is the most difficult to satisfy. However, $\hat{P}$ is included in the functional because it has the property that if a matrix, $P$, is nearly idempotent (i.e., the eigenvalues are between $-0.5$ and $1.5$), $\hat{P}$ is closer to idempotency. Thus, the density matrix obtained at the functional's minimum meets all of the criteria of the converged density matrix: normalization, idempotency, and commutation between $F$ and $P$.

We have found that in most cases it is necessary to apply additional purification transformations after each CGDMS step to maintain idempotency. However,
the purification transformations only improve the idempotency of matrices whose eigenvalues are between −0.5 and 1.5, so runaway solutions in which the density matrix loses idempotency are in principle possible.

In our implementation, we use the initial guess $P$ described in Chapter 3 formed from a set of localized molecular orbitals based on a Lewis dot structure of the molecule[28]. The density matrix is updated every SCF cycle using CGDMS. Early on in the SCF cycles $F$ is far from being self consistent, so it is not necessary that $P$ be obtained to a high precision degree. On the average, I found that only 2 to 4 CGDMS cycles are needed every SCF iteration to produce converged results in about the same number of SCF cycles as diagonalization. Thus, a typical CGDMS calculation requires about 28 matrix multiplications per SCF iteration[47].

As demonstrated in Chapter 2, when sparse matrix techniques are used, all of the linear algebra operations within CGDMS scale linearly with system size for insulating systems. To increase the efficiency of the calculation, a “fixed” matrix form is used within the CGDMS portion of the SCF calculation, where the matrix form is defined by the positions of the nonzero elements above a certain neglect threshold within the matrix[1]. To allow sufficient flexibility for matrix expansion, this matrix form is determined every SCF iteration by the combined forms of $F$, $P$, $PF$ and $FP$. This enhanced matrix form increases the size of the Fock and density matrices by adding zeros to positions within the matrices where normally no elements would be stored. Matrices within the CG cycles are forced to maintain this matrix form throughout the matrix operations. Calculations using this “fixed” matrix form obtain the same energy as obtained using “let-it-grow” (LIG) matrix
routines which allow the matrix form to change during each matrix operation, but are significantly faster. In my experience using this "fixed" form for matrix multiplication reduces the time cost by more than a factor of two. In addition, since CGDMS involves only linear algebra routines which are easily parallelizable such as matrix multiplication, my implementation of CGDMS is parallel.
Chapter 7

Linear Scaling Geometry Optimization of Plasminogen

Introduction

As an example of the use of CGDMS for a biologically interesting system I performed a gas-phase geometry optimization on the 1226 atom kringle 1 of plasminogen using PM3[8] implemented with CGDMS. Plasminogen is a protein present in human blood plasma and is a key component in the fibrinolytic mechanism. It is also thought to play a role in tissue repair, malignant transformation, macrophage function, ovulation, and embryo implantation. This is, to the best of our knowledge, the largest protein to be optimized semiempirically using a method other than DAC to achieve linear scaling of the computational time with system size.

Background

Until recently, large biomolecules such as proteins were outside of the computational reaches of quantum chemical calculations. However, with the development of linear scaling semiempirical methods throughout the 1990's, it is now possible to compute energies and gradients for molecules containing thousands of atoms. These accomplishments make possible semiempirical geometry optimizations on
large biological molecules, such as proteins. Most of the semiempirical geometry optimizations performed on proteins thus far have been carried out using the divide and conquer (DAC) approach to reduce the time required for the diagonalization step to linear scaling with the number of atoms. Lewis and coworkers performed a geometry optimization on a 1330 atom model for the cytidine deaminase active site[48]. They utilized the semiempirical PM3 Hamiltonian[8] with DAC to obtain linear scaling. For increased speed, the protein backbone was held fixed during the calculation. Later, Vincent, Dixon and Merz also used a linear scaling PM3 semiempirical program with DAC to carry out a full optimization on the geometry of the 1960 atom hen egg white lysozyme[49]. One difficulty with DAC methods is that the errors introduced by replacing diagonalization with DAC are difficult to control. Even though the DAC approximation is exact when the subsystem sizes reach the size of the entire system, the error is not a simple function of the subsystem size. Thus, it is difficult to predict the accuracy of a DAC calculation without also performing a calculation using diagonalization.

Therefore, if other linear scaling methods for replacing diagonalization could be found which only contain parameters in which the error as compared with the diagonalization result is predictable, they might be more appropriate candidates for performing geometry optimizations on large biological molecules. The methods mentioned earlier in this work which meet this requirement are CGDMS, pseudo-diagonalization, purification of the density matrix, and the Chebyshev expansion method. In these linear scaling diagonalization replacements, the introduced errors are easily controlled by changing simple thresholds. The errors depend on these
thresholds in a way that is, for the most part, system independent or is otherwise predictable. As the thresholds approach zero, the methods become exact. Thus, these methods might be more desirable than DAC for use in large molecule optimizations. The first such semiempirical geometry optimization on a protein was carried out by Stewart who optimized the geometry of a 740 atom crambin molecule using PM3 with PD[50]. In this work we present a gas phase geometry optimization on the 1226 atom kringle 1 of plasminogen using PM3 implemented with CGDMS.

**Methods**

The geometry optimization was carried out using the linear scaling PM3 code within a developmental version of the Gaussian suite of programs[51]. The diagonalization step of the SCF calculation is replaced by CGDMS. In this calculation, all matrix elements below the neglect threshold of $1 \times 10^{-3}$ au are discarded. Also, a distance cutoff of 15 Å is used to determine the form of the Fock matrix. These parameters are set such that an accuracy in energy as compared to the diagonalization result of about 0.05 kcal/mol is ensured.

The geometry was updated via an $O(N^2)$ scaling rational function optimization (RFO) technique (developed from the regular RFO method by Farkas and Schlegel[52]) combined with a modified version of the direct inversion in the iterative subspace procedure (GDIIS) (introduced by Császár and Pulay[53] and modified by Farkas[54]) using redundant internal coordinates. This optimization technique has been found to converge the geometries of large systems with about the same number of steps as the regular quasi-Newton based methods, but diagonalization
of the Hessian matrix via an $O(N^3)$ scaling operation is not required. The GDHS procedure uses the information from the previous points; therefore, it can quickly recover after steps resulting in higher energy and large forces. The coordinate transformations of the gradients and geometry updates between redundant internal and Cartesian coordinates were performed via a very fast $O(N^2)$ scaling algorithm introduced by Farkas and Schlegel[55]. Even though the geometry update step scales as $O(N^2)$ with system size, the CPU time it requires in the plasminogen calculation is small compared to the energy and gradient update CPU time as discussed below.

Starting coordinates for kringle 1 of plasminogen were obtained from the Protein Data Base[56]. Hydrogen atoms were added to the structure using GaussView[57], with the resulting geometry relaxed using the MM3 force field. This structure was used for input for the PM3 optimization using CGDMS as a replacement for diagonalization.

**Results and Discussion**

All calculations were carried out on a single MIPS R10K/195 MHz processor of a SGI Origin2000 computer. When optimizing large molecules such as proteins it is not entirely clear at which point to stop the geometry optimization. The defaults in Gaussian for geometry optimization convergence criteria are the following: RMS gradient = 0.36 kcal/mol/Å, maximum gradient = 0.53 kcal/mol/Å. RMS displacement = 0.0012 au. and maximum displacement = 0.0018 au, where the displacement is the Cartesian displacement measured in Bohrs. However, proteins contain large, floppy chains, causing the Cartesian displacement to be quite
large with even small changes in angles and dihedrals. We decided that a better convergence criterion for this optimization would be to use the change in internal coordinates for the system instead of the cartesian displacement. Also, for such a large system very small changes in bond lengths and angles should not be as important as in smaller molecules. Thus, it is not necessary to determine the geometry of the structure to such a high accuracy. We decided to loosen the convergence thresholds for this optimization to: RMS gradient = 3.6 kcal/mol/Å, maximum gradient = 5.3 kcal/mol/Å, and RMS internal coordinate displacement = 0.012 au (Bohrs/radians).

Using these criteria, and starting from the MM3 geometry, the PM3 optimization converged in 362 geometry updates, much faster than the number of steps reported by Stewart for the 740 atom crambin molecule (over 2000 cycles) and Vincent et. al for the 1960 atom hen egg white lysozyme (1023 cycles). However, these numbers of cycles are not directly comparable because the convergence criteria differ between calculations. Also, different proteins are used in each case, so the starting structures of the various proteins are of different degrees of accuracy, and the geometries themselves have different convergence properties. Our calculation confirms the findings of Stewart and Vincent that a large number of steps is required to optimize the geometry of proteins. Nevertheless, the number of steps in our calculation is about one third the number of atoms. The average time for the energy and gradient update is 64.8 CPU minutes and for the geometry update is 9.8 CPU minutes. Thus, the entire calculation required about 18.8 CPU days to complete.
The RMS gradient, as shown in Figures 4 and 5, oscillates quite wildly as the optimization progresses. This trend was also found in previous calculations[49, 50]. However, these fluctuations become smaller in size as the optimization proceeds, so that near the end of the calculation (in the last 20 steps), the maximum oscillations in the RMS gradient are no larger than 1 kcal/mol/A.

Figures 6 and 7 show the RMS internal coordinate displacement as a function of the geometry update step. As with the forces, the RMS displacements fluctuate wildly with the optimization step. It can also be noted that the displacements decrease very slowly as the optimization progresses. This is explainable by the large number of degrees of freedom in the system. Furthermore, the displacements are quite large throughout the calculation because the potential energy surface is very flat. Thus, large displacements cause very small changes in energy.

Figures 8 and 9 show the heat of formation as a function of the geometry update steps. As in Ref[50], the heat of formation quickly decreases during the first few geometry updates. Then it gradually levels off and decreases slowly in the final iterations. However, the energy does not decrease monotonically, but rather contains spikes as it decreases. This is probably an effect of using a different optimizer than the one utilized in Ref[50]. These spikes do not adversely affect the optimization, for the general trend in the energy is to decrease monotonically if the spikes are discarded. In the last several steps of the optimization, the spikes in energy disappear. Thus, over the last 15 geometry updates the change in the heat of formation is only $-0.07$ kcal/mol.
The final, optimized structure has a RMS deviation from the crystal structure of 2.4 Å as measured by Quanta[58]. Perhaps the most significant reason for this large deviation in structure is that the PM3 calculation was carried out in the gas phase, whereas the x-ray structure was determined from protein molecules along with several water molecules in a crystalline form. The exclusion of the effects of solvation on the surface of the protein may tend to cause the molecule to expand somewhat.

Several calculations on plasminogen were carried out to optimize the parameters in the geometry optimization step (such as the maximum step size). These calculations all used the same starting geometry (with an energy 1300 kcal/mol above the energy at the optimized geometry), but contained different optimization parameters. The final geometries of each of these calculations differed from one another slightly. The RMS deviations between their geometries were at most 0.9 Å, and their energies differed by less than 6 kcal/mol. This occurred because the molecule is very floppy, causing the potential energy surface to be very flat. Thus, each optimization obtained different local minima on the potential energy surface.

Conclusions

CGDMS in conjunction with the GDIIS/RFO optimizer is an effective tool for semiempirical geometry optimizations of proteins. We found that the number of geometry update steps required by the GDIIS/RFO optimizer is roughly equal to a third of the number of atoms in the system. Thus, with linear scaling energy and gradient updates, such as PM3/CGDMS, calculations on proteins containing over
a thousand atoms are well within the reach of current computational resources. We expect that protein geometry optimizations using semiempirical methods will soon become routine. Also, since proteins are naturally in a solvated environment, it is important to add solvation effects to the calculations. Work along these lines is in progress.
Figure 4  RMS gradient (kcal/mol/A) is shown as a function of the geometry optimization step.[4]
Figure 5  RMS gradient (kcal/mol/Å) is shown as a function of the geometry optimization step for the final portion of the optimization. [4]
Figure 6  RMS internal coordinate displacement (Bond lengths in Bohrs and angles in radians) is shown as a function of the geometry optimization step.[4]
Figure 7  RMS internal coordinate displacement (Bond lengths in Bohrs and angles in radians) is shown as a function of the geometry optimization step for the final portion of the optimization.[4]
Figure 8  Heat of formation (kcal/mol) is shown as a function of the geometry optimization step.[4]
Figure 9  Heat of formation (kcal/mol) is shown as a function of the geometry optimization step for the final portion of the optimization. [4]
Chapter 8

Convergence Properties of Conjugate Gradient Density Matrix Search

Introduction

In addition to the conjugate gradient density matrix search (CGDMS) method being useful for producing linear scaling in semiempirical methods, it proves to have superior self-consistent field (SCF) convergence properties in cases where the SCF does not normally converge using diagonalization. Self-consistent field (SCF) convergence failures are a significant problem when performing \textit{ab initio} calculations on molecules using methods such as Hartree Fock (HF) or density functional theory (DFT). These problems occur when oscillations or divergence sets in during the SCF procedure and are especially prominent in calculations on molecules containing transition metals, actinides, and other heavy metals, and systems containing small gaps between their highest occupied molecular orbitals (HOMO) and lowest unoccupied molecular orbitals (LUMO). In fact, a formal relationship is derived in Ref([59]) connecting small HOMO–LUMO gaps with poor SCF convergence.

In the past, several techniques have been proposed which improve SCF convergence in cases which do not normally converge[60, 61, 62, 63, 64, 65, 66, 67]. One distinct possibility is to improve the quality of the density matrix guess.
The Hückel guess density matrix which is widely used as a starting point for \textit{ab initio} calculations describes metal–ligand interactions poorly. When a guess which describes the charge distribution more accurately is provided, SCF convergence is greatly enhanced\cite{60}. Seeger and Pople introduced a technique which improves SCF convergence by applying a univariate energy minimization during each SCF iteration\cite{61}. A third convergence enhancement technique is to increase the molecule's HOMO–LUMO gap by shifting the virtual orbitals to higher energies (NShift)\cite{62}. This prevents swapping from occurring between the HOMO and LUMO thus avoiding oscillations between one SCF cycle to the next. In other cases, fractionally occupying orbitals around the Fermi level has been found to enhance SCF convergence\cite{63,64,65}. Willetts and Handy proposed that alternatives to diagonalization can also be used to improve SCF convergence\cite{66}. They found that by replacing diagonalization in the SCF with a molecular dynamics wavefunction propagation scheme, SCF convergence can be achieved even when molecules are at highly distorted geometries\cite{66}.

The number of cycles required for SCF convergence using many of the above methods can be reduced by extrapolation of the Fock matrix in a procedure called direct inversion of the iterative subspace (DIIS)\cite{68,69}. To reduce the commutation error between the density and Fock matrices, the Fock matrix is written as a linear combination of the current Fock matrix and a number of Fock matrices from previous SCF iterations. DIIS greatly improves SCF convergence when combined with normal diagonalization, level shifting, and fractional occupation techniques. Sellers\cite{67} developed a method in which he modified DIIS to extrapolate in the
optimized subspace of a univariate energy minimization technique similar to that proposed by Seeger and Pople. He found that this method has fast SCF convergence for difficult cases[67]. Because of its ability to accelerate SCF convergence, we will use DIIS in combination with all of our calculations in this work.

We propose yet another technique for improving SCF convergence. Conjugate gradient density matrix search (CGDMS) has been used to replace the $\mathcal{O}(N^4)$ scaling diagonalization step to produce linear scaling methods for calculations on large molecules[1, 3, 34]. In this work we introduce CGDMS as an alternative method to obtain SCF convergence in systems which fail to converge using diagonalization with level shifting and DIIS. A similar approach has also been attempted by another group[70].

CGDMS has several properties which may contribute to better SCF convergence than diagonalization methods have. Since this method does not directly involve forming molecular orbitals, it has no direct band gap. Thus, the convergence properties of the SCF procedure may be completely different when diagonalization is replaced by CGDMS. Another CGDMS property which may be important to SCF convergence is the fact that the functional need not be fully minimized every SCF iteration, but can be "progressively converged"[71]. The minimization is stopped prematurely early on in the SCF iterations. In other words, the CGDMS iterations are stopped before the commutator of the Fock and density matrices is fully zero ($[F, P] \neq 0$). However, the idempotency and normalization properties are enforced after each CGDMS step, so these are fully satisfied during each SCF iteration. As the calculation progresses, the density matrix becomes closer and closer to the
converged density matrix. until at the final iteration the density matrix is fully converged. Thus, during a CGDMS calculation the density matrix changes more gradually than in a calculation utilizing diagonalization, allowing a more smooth convergence to self-consistency. This property lends itself well to CGDMS's potential use as a SCF convergence enhancer in cases where the energies oscillate wildly as the SCF iterations proceed. This work gives some examples of calculations in which replacing diagonalization with CGDMS improves the SCF convergence.

Computational Details

All calculations were carried out using a developmental version of the Gaussian suite of programs[12]. Both the CGDMS and the diagonalization calculations were performed using the exact same parameters and code unless otherwise specified. The initial guess density matrix used was the "Projected New EHT Guess"[12]. A DIIS extrapolation scheme was used which forms the current Fock matrix as a linear combination of the previous 10 Fock matrices, using a cyclical queue. When specified, the energies of the virtual orbitals were shifted using the density matrix \((P)\) from the previous SCF iteration according to the following prescription:

\[
F_{\text{shifted}} = F + V_{\text{Shift}} \times (I - P_{i-1})
\]  

(8.1)

where \(V_{\text{Shift}}\) (units of Hartrees) is the amount of energy the virtual orbitals are shifted. SCF convergence was said to be achieved when the root mean square difference between the current and previous density matrices was \(2.0 \times 10^{-7}\) a.u. and the maximum difference was \(2.0 \times 10^{-5}\).
Several parameters were used in the CGDMS calculations. A maximum step size of 0.1 a.u. was used to prevent the density matrix from changing so rapidly that it would become non-idempotent. Three McWeeny purification transformations [Eq(2)] were performed after each conjugate gradient step. These are necessary to restore idempotency since the search is not explicitly constrained to be idempotent. To implement progressive convergence of the density matrix, the conjugate gradient cycles were stopped after only 4 cycles. Thus, the commutator between the Fock and density matrices was not allowed to fully converge to zero early on in the SCF iterations, but was gradually decreased as the SCF progressed until, at SCF convergence, the matrices commuted.

The results found in this chapter are not directly comparable with those found in Ref([63]) because the DIII schemes used differed in their methods and criteria for removing old matrices from the extrapolation. However, for consistency the same DIII code was used for both the CGDMS and the diagonalization calculations in this work.

The molecules considered in this chapter and their geometrical parameters are shown in Table 1. Because CrC, Cr₂, and UF₁ all have low lying triplet states, both their singlet and triplet states were calculated. The methods used for the calculations were the local density approximation, LSDA, the gradient-corrected density functional BPW91, and the hybrid functional B3LYP. The molecules containing uranium were calculated with the LANL2DZ basis set which contains effective core potentials, while the rest were calculated using the 3-21G and 6-311G** basis sets.
Also, even though these molecules all contain symmetry, the calculations were performed without symmetry.

The energies of the various types of calculations at the same levels of theory and basis sets were compared with one another. In addition, the calculations were tested for Aufbau principle violations by comparing the converged density matrix of the VShifted or CGDMS calculations with a density matrix formed by diagonalizing the non-level shifted converged Fock matrix.

**Results and Discussion**

Table 2 shows the number of cycles required to converge the SCF in CGDMS and diagonalization single point energy calculations on CrC as a function of the VShift. The number of SCF cycles needed for the CGDMS calculation increases from 469 cycles for no VShift, to 592 cycles for a VShift of 0.2 a.u. On the other hand, the number of SCF cycles required to converge the diagonalization calculation is reduced from >901 for no VShift to 244 when a VShift of 0.2 a.u. is applied. Although the effect of introducing a VShift in diagonalization calculations is varied, the trend shown when using CGDMS is reproduced in most of the calculations performed. Thus, all CGDMS calculations are performed without a VShift, whereas diagonalization calculations are performed with and without VShifts.

The number of SCF cycles required to obtain self-consistency in calculations on four molecules which we deem to be representative of cases which are difficult to converge is presented in Table 3. We found that a significant portion of level shifted and CGDMS calculations converged to excited states and/or states con-
taining Aufbau principle violations. In this chapter, a calculation is said to have converged to an excited state if it has a higher energy than other results using the same method and basis set. It is not possible to reliably determine whether a calculation is converged to a ground state with the above method, so we also check for Aufbau principle violations (holes below the Fermi level) in the converged density matrix. However, even results containing Aufbau principle violations are not necessarily excited states. There are cases with DFT in the literature, such as C$_2$, in which the lowest energy state does not satisfy the Aufbau principle\cite{72}.

Results containing Aufbau principle violations are possible when a VShift is applied and one or more occupied orbitals are switched with virtual orbitals in the density matrix used as a guess at the beginning of the calculation. When this happens, the VShift is added to the misplaced occupied orbital(s) instead of the virtual one(s). Throughout the calculation, the correct ordering of orbitals is not restored because the misplaced occupied orbital is pushed too high in energy by the VShift to be able to drop back to the ground state. To lessen the likelihood of this outcome, a small VShift must be used. In our experience VShifts smaller than 0.1 Hartrees do not increase convergence significantly and those much larger than 0.2 Hartrees quite often lead to excited states.

Aufbau principle violations are also possible in results calculated with CGDMS. If there are holes below the Fermi level in the guess density matrix, they are sometimes preserved throughout the CGDMS calculation resulting in an excited state. As an example, we performed a Hartree Fock/6-31G(d,p) calculation on N$_2$. The occupations of the initial guess orbitals were altered such that the two degenerate
orbitals above the Fermi level ($\Pi_\tau$ symmetry) were occupied instead of the two degenerate orbitals below the Fermi level ($\Pi_u$ symmetry). Throughout the CGDMS calculation, this occupation arrangement was preserved and the calculation converged to an excited state 0.5 Hartrees higher in energy than the ground state. However, when other alterations in the orderings of the guess orbitals were tested, the CGDMS calculation converged to the ground state.

Because CGDMS sometimes follows surfaces which contain Aufbau principle violations throughout the entire calculation, we attempted to converge the calculations to a ground state by using different density matrices as initial guesses. One method might be to use an initial guess density matrix containing several interchanged occupied and virtual orbitals which correspond to the swapped orbitals in the calculation which converged to the non-Aufbau state. For example, the BPW91/3–21G calculation on $^1$CrC converged to a state in which the HOMO–2 orbital was switched with the LUMO orbital. We altered the guess density matrix occupations by interchanging these two orbitals and ran the CGDMS calculation as normal. This calculation converged to a lower energy than the non-altered calculation, but still contained holes below the Fermi level corresponding to the HOMO–1 orbital switched with the LUMO+1 orbital. Another method may be to start the problematic calculation using the density matrix of a CGDMS calculation on a method which converged to the ground state. We tried this for the CrC BPW91/3–21G calculation using the converged B3LYP/3–21G density matrix as an initial guess. This calculation also converged to the state in which the HOMO–2 orbital was switched with the LUMO orbital. A third method might be to restart
the calculation using as a guess the result from the first calculation with the guess density matrix rearranged such that the Aufbau principle is satisfied. We tried starting the CrC calculation from both of the states mentioned above, but each time the calculation resulted in a state where the HOMO−1 orbital was switched with the LUMO+1 orbital.

Another observation from the calculations in Table 3 is the convergence properties of CGDMS as a function of the multiplicity of the molecule. Molecules in the singlet state tend to converge more often or much faster than those in the triplet state. This is despite the fact that the triplet state of the molecules calculated here is the more energetically stable state. For example, singlet CrC calculations all converge in about one half to one fourth of the number of steps required for the corresponding triplet calculations. Likewise, some of the triplet Cr₂ and UF₄ calculations do not converge in the allotted number of SCF cycles even though all of the singlet calculations did converge.

We now compare the results of the CGDMS calculations with diagonalization calculations appearing in Table 3. CGDMS found SCF convergence for 29 of the 33 calculations, while diagonalization without an applied VShift found SCF convergence for only 15 of these calculations. However, when only the singlet states are considered, CGDMS found convergence in all of the 18 cases while diagonalization found convergence in only 5 of these cases. Thus, CGDMS does very well in finding some state (not necessarily the ground state) in singlet calculations as compared to diagonalization even when there are lower triplet states. When level shifting was used with diagonalization, many more of the diagonalization calculations converged.
In fact, when considering calculations using all three values of VShifts ranging from 0.0 to 0.2 Hartrees, all but six of the calculations which had converged with CGDMS also converged with diagonalization, as well as several of the triplet state calculations which did not converge with CGDMS. However, the convergence properties of the calculation are not a predictable function of the VShift. Some calculations which converged with a small or no VShift did not converge with a large VShift, and vice versa. It is probable that more of these calculations could be converged by varying the value of the VShift, but this is a trial-and-error process which must be performed independently for each molecule.

It is also important to look at whether the calculations converged to a ground state. Because it is difficult in practice to determine whether or not a calculation converged to its ground state, for the sake of discussion, instead of the ground state, we will consider the lowest energy result in each set of calculations, even though it may contain Aufbau violations. There were 15 calculations in which both diagonalization with some value of level shifting and CGDMS converged to the same lowest energy states. Three additional calculations converged to a lowest energy state using level shifted diagonalization which did not converge with CGDMS. However, CGDMS converged 12 calculations to a lowest energy state which level-shifted diagonalization was not able to converge. One point worth mentioning is that eight of these 12 calculations converged to states with Aufbau principle violations. This is in part due to the CGDMS's ability to remain on one electron configuration surface, while the diagonalization calculations tend to oscillate between electron configurations hindering convergence.
Also of interest is the number of SCF cycles required to converge the density matrix. With several exceptions when both CGDMS and diagonalization converged to the same state, the CGDMS calculations tended to converge at slower rates than the diagonalization calculations. One possible reason for this is that diagonalization calculations produce a density matrix which commutes with the current Fock matrix during each SCF iteration, whereas the CGDMS calculation slowly converges the density matrix throughout the calculation until SCF convergence, the density and Fock matrices commute. This progressive convergence is helpful when calculations have difficulty converging due to rapid oscillations in energy and a "smoothing" is needed, but it reduces the convergence rate when regular diagonalization calculations converge rapidly to the correct answer. Another factor causing slow SCF convergence in CGDMS calculations is that it often takes many SCF cycles to go from one set of orbital occupations to another. When a CGDMS calculation begins with holes below the Fermi level, it sometimes goes through several different sets of orbital occupations until it finds the final orbital ordering. This process requires many SCF cycles.

Conclusions

Our investigation shows that CGDMS is an interesting alternative to diagonalization for converging troublesome molecules to ground states. The SCF convergence properties of CGDMS are comparable with those of diagonalization with level shifting, while CGDMS does not have the arbitrary parameter which must be varied when using VShifts. However, it must be noted that when using CGDMS or level
shifting it is possible to converge to an excited state, so it is important to con-
firm that the converged result is the one desired. In conclusion, the results in this
chapter demonstrate CGDMS’s role as a promising alternative for handling difficult
SCF convergence problems.

Table 1  Geometric parameters of CrC, Cr₂, UO₂(OH)₁ and UF₄ with
bond lengths given in Å and angles given in degrees.[73]

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrC</td>
<td>Cr - C</td>
<td>2.00</td>
</tr>
<tr>
<td>Cr₂</td>
<td>Cr - Cr</td>
<td>2.00</td>
</tr>
<tr>
<td>UO₂(OH)₁ᵃ</td>
<td>U - O</td>
<td>1.84</td>
</tr>
<tr>
<td></td>
<td>U - (OH)</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td>O - H</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>O - U - O</td>
<td>180.0</td>
</tr>
<tr>
<td></td>
<td>(HO) - U - (OH)</td>
<td>90.0</td>
</tr>
<tr>
<td></td>
<td>(U - O - H)</td>
<td>106.1</td>
</tr>
<tr>
<td></td>
<td>(O - U - O - H)</td>
<td>0.0</td>
</tr>
<tr>
<td>UF₄ᵇ</td>
<td>U - F</td>
<td>1.98</td>
</tr>
</tbody>
</table>

ᵃOH ligands opposite one another have their H’s pointing in the same direction
and ligands 90° from one another have their H’s pointed in opposite directions
from one another.
b UF₄ is in a tetrahedral geometry.
Table 2  Number of cycles needed for SCF convergence as a function of the $V_{\text{Shift}}$ for CGDMS and diagonalization BPW91/3-21G single point energy calculations on the $^1\text{Cr}_2$ molecule.[73]

<table>
<thead>
<tr>
<th>Method</th>
<th>$V_{\text{Shift}}$ (Hartrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>Diagonalization</td>
<td>X*</td>
</tr>
<tr>
<td>CGDMS</td>
<td>469</td>
</tr>
</tbody>
</table>

*aNo convergence in 900 cycles.
Table 3  Number of cycles needed for SCF convergence for several molecules using CGDMS and diagonalization with a VShift (VS) of 0.0, 0.1 and 0.2 Hartrees. DIIS is used for all the calculations.[73]

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Method</th>
<th>Diagonalization</th>
<th>(\text{VS=0.0} )</th>
<th>(\text{VS=0.1} )</th>
<th>(\text{VS=0.2} )</th>
<th>CGDMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(^1\text{CrC} )</td>
<td>LS/DA/3-21G</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>106(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LS/DA/6-311G**</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>56(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/3-21G</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>67(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/6-311G**</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>71(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/3-21G</td>
<td>72(^c)</td>
<td>X(^a)</td>
<td>76(^c)</td>
<td>65(^c)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/6-311G**</td>
<td>23(^c)</td>
<td>52(^c)</td>
<td>30(^c)</td>
<td>51(^c)</td>
<td></td>
</tr>
<tr>
<td>(^3\text{CrC} )</td>
<td>LS/DA/3-21G</td>
<td>X(^a)</td>
<td>403(^c)</td>
<td>210(^c)</td>
<td>395(^c)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LS/DA/6-311G**</td>
<td>X(^a)</td>
<td>41(^c,d)</td>
<td>X(^a)</td>
<td>146(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/3-21G</td>
<td>184(^c)</td>
<td>36(^c,d)</td>
<td>179(^d)</td>
<td>416(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/6-311G**</td>
<td>41(^c)</td>
<td>49(^c,d)</td>
<td>162(^d)</td>
<td>116(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/3-21G</td>
<td>29(^c)</td>
<td>X(^a)</td>
<td>15(^c)</td>
<td>197(^c)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/6-311G**</td>
<td>40(^c)</td>
<td>31(^c)</td>
<td>42(^c)</td>
<td>108(^c)</td>
<td></td>
</tr>
<tr>
<td>(^1\text{Cr}_2 )</td>
<td>LS/DA/3-21G</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td>392(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LS/DA/6-311G**</td>
<td>X(^a)</td>
<td>36(^c,d)</td>
<td>24(^c,d)</td>
<td>271(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/3-21G</td>
<td>44(^c)</td>
<td>244(^c)</td>
<td>469(^d)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/6-311G**</td>
<td>X(^a)</td>
<td>22(^c,d)</td>
<td>33(^c,d)</td>
<td>232(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/3-21G</td>
<td>121(^c)</td>
<td>197</td>
<td>201(^c)</td>
<td>353(^c)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/6-311G**</td>
<td>49(^c)</td>
<td>17(^c)</td>
<td>23(^c)</td>
<td>102(^c)</td>
<td></td>
</tr>
<tr>
<td>(^1\text{Cr}_2 )</td>
<td>LS/DA/3-21G</td>
<td>23(^c)</td>
<td>99(^c)</td>
<td>330(^c)</td>
<td>X(^a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LS/DA/6-311G**</td>
<td>68(^c)</td>
<td>63(^c)</td>
<td>196(^c)</td>
<td>482(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/3-21G</td>
<td>25(^c)</td>
<td>416</td>
<td>213(^c)</td>
<td>457(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/6-311G**</td>
<td>57(^c)</td>
<td>71(^c)</td>
<td>X(^a)</td>
<td>306(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/3-21G</td>
<td>25(^c)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/6-311G**</td>
<td>74(^c)</td>
<td>332(^c)</td>
<td>X(^a)</td>
<td>X(^a)</td>
<td></td>
</tr>
<tr>
<td>(^1\text{UO}_2(\text{OH})_4 )</td>
<td>LS/DA/LANL2DZ</td>
<td>X(^a)</td>
<td>33(^c)</td>
<td>X(^a)</td>
<td>30(^c)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/LANL2DZ</td>
<td>X(^a)</td>
<td>35(^c)</td>
<td>X(^a)</td>
<td>29(^c)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/LANL2DZ</td>
<td>31(^c)</td>
<td>29(^c)</td>
<td>31(^c)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(^1\text{UF}_4 )</td>
<td>LS/DA/LANL2DZ</td>
<td>X(^b)</td>
<td>277(^c)</td>
<td>X(^b)</td>
<td>557(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/LANL2DZ</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td>201(^c)</td>
<td>193(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/LANL2DZ</td>
<td>X(^b)</td>
<td>197(^c)</td>
<td>100(^c)</td>
<td>785(^c)</td>
<td></td>
</tr>
<tr>
<td>(^3\text{UF}_4 )</td>
<td>LS/DA/LANL2DZ</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td>913(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BPW91/LANL2DZ</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td>98(^d)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B3LYP/LANL2DZ</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td>X(^b)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(\text{a} \) No convergence in 501 cycles.

\(\text{b} \) No convergence in 1001 cycles.

\(\text{c} \) Converged to a higher energy than similar calculations.

\(\text{d} \) Converged density matrix has Aufbau principle violation.
Chapter 9

Pseudodiagonalization

Unlike conjugate gradient density matrix search (CGDMS) which searches for the density matrix directly, pseudodiagonalization (PD) performs pairwise orthogonal rotations of the molecular orbitals similar to Jacobi rotations, from which the density matrix can be obtained by

\[ P_{ij} = \sum_{k=1}^{N_a} \psi_{ik} \psi_{jk}. \]  

(9.1)

As mentioned in Chapter 5, PD was first developed by Stewart, Csiszár, and Pulay [42] and later adapted as a linear scaling method by Stewart [28]. PD is based on the fact that a necessary and sufficient condition for a stationary solution of the SCF equations is that \( F_{ov} = 0 \), where \( F_{ov} = C_a^T F C_v \). \( C_a \) are the occupied molecular orbitals and \( C_v \) are the virtual orbitals.

PD begins with a set of localized molecular orbitals based on the Lewis dot structure of the chemical system. During each SCF iteration a series of pairwise orthogonal rotations between the occupied and virtual orbitals based on first-order perturbation theory is performed which reduces the value of the non-zero elements of \( F_{ov} \) [42]. If the occupied orbitals are given by the index, \( i \), and the virtual orbitals are given by the index, \( a \), these rotations are given by the following equations:

\[ C'_i = \cos(\theta) \cdot C_i - \sin(\theta) \cdot C_a \]  

(9.2)
\[ C'_a = \sin(\theta) \cdot C_i + \cos(\theta) \cdot C_a \]  \hspace{1cm} (9.3)

where \( \theta \) is the angle of rotation found by the following method:

\[ A = < C_i | F | C_i > \]  \hspace{1cm} (9.4)

\[ B = < C_a | F | C_i > = < C_i | F | C_a > \]  \hspace{1cm} (9.5)

\[ C = < C_a | F | C_a > \]  \hspace{1cm} (9.6)

\[ \sin(\theta) = (A - C)^{-1} B \]  \hspace{1cm} (9.7)

\[ \cos(\theta) = (1 - \sin^2(\theta))^{1/2} \]  \hspace{1cm} (9.8)

Although all of the \( E_{ov} \) elements are not completely zeroed out during each SCF iteration, PD works successfully because it is not necessary to obtain the exact eigenvectors of \( F \) every SCF iteration. The molecular orbitals need only be sufficiently converged to bring \( F \) closer to SCF convergence. In practice, it is sometimes helpful to iterate PD between 1 to 4 times within the SCF cycle to reduce the total number of SCF cycles needed to reach self-consistency[3]. If the orbital rotations are counted as a single matrix multiply, a typical PD calculation requires about 10 matrix multiplies per SCF iteration[74]. Unlike with CGDMS, LIG style multiplications are used in PD because the forms of the matrix products are not known before the multiplication.

PD scales linearly with system size because the molecular orbitals normally localize, usually to around one hundred atoms for the largest systems that we (and others[28]) have tried. Also, the number of elements in \( E_{ov} \) increases linearly with system size. Thus, since the number of orthogonal rotations increases linearly with system size and the number of elements in each MO is constant with system size.
PD is a linear scaling method. A comparison of this method with other linear scaling techniques will be given in Chapter 12. Also, like CGDMS, PD is easily parallelized so my implementation is parallel.
Chapter 10

Chebyshev Expansion Method

Whereas conjugate gradient density matrix search and pseudodiagonalization require an initial density matrix or set of molecular orbitals to be updated each SCF iteration, the Chebyshev expansion method (CEM) calculates the density matrix, $P$, directly using only the Fock matrix, $F$, and a Fermi–Dirac distribution for the orbital occupations[2, 3, 23, 24, 26]. Since $P$ can be defined using the Fermi–Dirac distribution,

$$P = \frac{1}{1 - e^{\beta(F-\mu)}}$$  \hspace{1cm} (10.1)

where $\beta$ is the (fictitious) inverse temperature and $\mu$ is the chemical potential. $P$ can be formed using a polynomial expansion of this equation. It has been found that this function is efficiently and accurately described by an expansion of Chebyshev polynomials[23]. Thus,

$$P = f[\beta(F-\mu)] = c_0 - \sum_{i=1}^{CPO} c_i T_i$$  \hspace{1cm} (10.2)

$$T_0(F) = I$$  \hspace{1cm} (10.3)

$$T_1(F) = F$$  \hspace{1cm} (10.4)

$$T_{j+1}(F) = 2FT_j(F) - T_{j-1}(F)$$  \hspace{1cm} (10.5)

where $f$ is a Fermi–Dirac distribution. We found that the Chebyshev polynomial order (CPO) required for a particular accuracy could be reduced by choosing $f$
to be the complementary error function instead of the Fermi Dirac distribution. Using this function, $\beta$ is a scaling factor similar to a fictitious temperature which determines how rapidly the function drops to zero at the Fermi level and is chosen such that $P$ is idempotent. Since Chebyshev polynomials require a domain between $-1$ and $1$, $\bar{F}$ is chosen to be $F$ scaled such that its eigenvalues are between these values:

$$\bar{F} = \frac{F - a}{b} \geq \{\epsilon_i\} \in [-1, 1].$$

(10.6)

where $a$ is the average of the highest and lowest eigenvalues and $b$ is half of the span of the matrix. The highest and lowest eigenvalues needed to scale $F$ are found via a linear-scaling Lanczos method[39]. $\mu$, the Fermi energy of the system, is determined by enforcing that $Tr(P) = N_{elec}$ using a combination of bisection and Newton-Rapson methods[2, 75]. $\{c_i\}$ are the Chebyshev coefficients derived from the Fermi-Dirac distribution, and $\{T_i\}$ are the Chebyshev recursion relations.

The CPO determines the accuracy of the density matrix approximation, where larger values of CPO result in increasing accuracy. The CPO required for a certain accuracy varies for different molecules since it depends on the band gap of the system[26]. mHartree accuracy is attained for calculations on polyglycine chains, water clusters and RNA molecules[76] using CPOs of 75, 60 and 100 respectively.

To reduce the number of matrix multiplications needed during each CEM step, the $\{T_i\}$ are calculated outside of the Newton-Rapson iterations[26]. The same expanded matrix form used for CGDMS is used to determine the form of $P$ in CEM. We found that this choice produces the same energy obtained using LIG matrix operations. Thus, CEM requires CPO plus two matrix multiplications each SCF
iteration to calculate $P[77]$. Like CGDMS, all linear algebra operations used by CEM scale linearly with system size when the number of matrix elements scales linearly with system size. Thus, CEM is a linear scaling replacement for diagonalization. Also, it contains matrix operations which are easily parallelized so my implementation of CEM is parallel.
Chapter 11

Purification of the Density Matrix

The idea behind the purification of the density matrix method (PDM) can be obtained by observing what happens when the fictitious temperature is increased in the Chebyshev expansion method (CEM). When this is done, two things happen. First, the Chebyshev polynomial order (CPO) required to model the Fermi–Dirac distribution is reduced. This is because as the temperature is increased, the rapid decrease in the Fermi–Dirac distribution around the Fermi level from a functional value of 1 to a value of 0 broadens and becomes smoothed out. Thus, the number of Chebyshev polynomials required to represent this function decreases. In fact, a formal relationship between the fictitious temperature and the CPO is derived in Ref[26]. Secondly, the resulting density matrix becomes nonidempotent. This is also a result of the "smoothing" of the distribution around the Fermi level—the occupied eigenvectors near the Fermi level have eigenvalues of less than one and the unoccupied eigenvectors have eigenvalues greater than zero. However, we know how to make a matrix which is nearly idempotent idempotent—by the McWeeny purification transformation. Thus, the same accuracy can be achieved with a smaller CPO in CEM if purification transformations are applied after the Chebyshev expansion. Table 4 shows the number of purification cycles needed to obtain mHartree accuracy on (Gly)_{100} calculations with a given CPO.
By adding purification transformations the CPO can be reduced to two, which is simply a linear combination of the identity matrix and $F$. This reduces the number of matrix multiplications needed to produce the same accuracy, increasing the speed of the calculation. The extreme of this method, with a CPO of two, is simply a variant of the method recently proposed by Palser and Malolopoulos which we will refer to as purification of the density matrix[25].

We also implemented the canonical purification method proposed by Palser and Manolopoulos[25]. Every SCF iteration, an initial guess density matrix ($P_0$) is formed:

$$P_0 = \lambda \left( \mu I - F \right) + \frac{N_{\text{elec}}}{N} I$$  \hspace{1cm} (11.1)$$

$$\lambda = \min \left\{ \frac{N_{\text{elec}}}{F_{\text{max}} - \mu}, \frac{N - N_{\text{elec}}}{\mu - F_{\text{min}}} \right\}$$  \hspace{1cm} (11.2)$$

$$\mu = \frac{tr[F]}{N}$$  \hspace{1cm} (11.3)$$

where $N$ is the number of basis functions, and $F_{\text{max}}$ and $F_{\text{min}}$ are the maximum and minimum eigenvalues of $F$. Our implementation of PDM differs from that of Palser and Manolopoulos in that instead of estimating $F_{\text{max}}$ and $F_{\text{min}}$ using upper and lower bounds of the eigenvalues through Gershgorin’s formulas, we find these values via a linear scaling Lanczos method. We found that the Lanczos implementation requires a factor of 1.5 to 2 fewer purification cycles than the Gershgorin implementation because the Lanczos method uses the exact maximum and minimum eigenvalues of $F$ instead of upper and lower bounds to them. The guess density matrix is then purified in such a way as to conserve the number of electrons. McWeeny’s purification function $f(x) = 3x^2 - 2x^3$ has an unstable fixed
point at \( x = c \) (where \( f(x) = x \) and \( f'(x) \geq 1 \)). By allowing \( c \) to vary between \( c = 0 \) and \( 1 \) instead of being fixed at \( c = 1/2 \), the number of electrons can be fixed.

The resulting purification transform is as follows:

\[
P_0 = \begin{cases} 
(1 - 2c_n)P_n + (1 + c_n)P_n^2 - P_n^3)/(1 - c_n) & \text{if } c_n \leq 1/2, \\
(1 + c_n)P_n^2 - P_n^3)/c_n & \text{if } c_n > 1/2 
\end{cases}
\]  
(11.4)

\[
c_n = \frac{\text{tr}[P_n^2 - P_n^3]}{\text{tr}[P_n = P_n^2]}  
\]  
(11.5)

We found that this purification scheme is more efficient than keeping \( c \) fixed as in McWeeny's purification transformation, and then scaling the purified \( P \) such that it contains the right number of electrons. The purification transformations are repeated until a measure of idempotency \( \left( \frac{\sqrt{\text{tr}(P_n^2) - \text{tr}(P_n^3)}}{\text{tr}(P_n - P_n^3)} \right) \) is below \( 1 \times 10^{-7} \), which we found sufficient for obtaining nHartree accuracy in energy. To determine the form of the matrices, we use the fixed matrix form described above for CGDMS. We found that a typical PDM calculation requires about 22 matrix multiplications per SCF iteration[78]. PDM is a linear scaling replacement for diagonalization because it only uses linear algebra operations which scale linearly with system size when the number of matrix elements scales linearly with system size. In addition, like the other linear scaling diagonalization replacements mentioned in this work, my implementation of PDM is parallel.
Table 4  The number of purification cycles required to produce mHartree accuracy in 100 polyglycine calculations with several CPO's. All of the calculations took 6 SCF iterations to converge but the calculation with a CPO of 30 which took 7 SCF iterations.[3]

<table>
<thead>
<tr>
<th>CPO</th>
<th>Number of Purifications</th>
<th>CPU Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>0</td>
<td>5.28</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>3.83</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>3.60</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>3.37</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>4.13</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>2.82</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>2.53</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2.52</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>2.53</td>
</tr>
</tbody>
</table>
Chapter 12

Comparison of Linear Scaling Methods

Introduction

Conjugate gradient density matrix search (CGDMS), pseudodiagonalization (PD), Chebyshev expansion method (CEM) and purification of the density matrix (PDM) all have a time cost which scales linearly with system size. So which one should we use? This chapter attempts to answer this question by giving a comparison of these methods, addressing the following issues: CPU scaling with system size, memory requirements, reliability and speed.

Computational Details

In this work, all calculations were performed using AM1 within the Gaussian suite of programs[79]. To determine the form of $F$ within the SCF procedure a 15 Å $F_{\mu\lambda}$ atom-atom cutoff is applied where $\mu$ and $\lambda$ are basis functions on different atoms[1]. The error in energy introduced by this cutoff in insulating systems is on the order of a few nHartrees from the full matrix result. Also, to produce linear scaling in the Coulomb portion of the calculation, a 25 Å $F_{\mu\nu}$ atom-atom cutoff is applied, where $\mu$ and $\nu$ are basis functions on the same atom. This cutoff has a "soft" tail which gradually scales the contributions to zero reducing the resulting
error. The soft $F_{\mu\nu}$ cutoff produces results which are in mHartree agreement in energy from calculations without this cutoff[1]. However, since the $O(N^2)$ scaling contribution to the $F_{\mu\nu}$ elements has a very small prefactor, the $F_{\mu\nu}$ cutoff is only needed for the very largest of the molecules studied here. It produces only negligible speedup in calculations on molecules containing up to a few thousand atoms. To ensure mHartree accuracy in energy, $F$ and $P$ are pruned with a neglect threshold of $5 \times 10^{-3}$ a.u. for PD and CGDMS. A slightly smaller neglect threshold of $1 \times 10^{-5}$ a.u. was required for CEM calculations to converge in a reasonable number of SCF iterations (around 10 to 20).

Since the density matrix is known to converge more slowly than the energy, our convergence criteria are based on both convergence of the energy and the density matrix. The energy is converged to at least $5 \times 10^{-4}$ Hartrees. Two properties of a converged density matrix are that it is idempotent, and it commutes with $F$. Thus, we consider the density matrix to be converged when all of the following criteria are below $1 \times 10^{-4}$ a.u.: RMS convergence $\left( \frac{\sqrt{\text{Tr}(P_{\text{old}} - P_{\text{new}})^2}}{\text{Tr}(P)} \right)$, idempotency $\left( \frac{\sqrt{\text{Tr}(P^2 - P)^2}}{\text{Tr}(P)} \right)$, and commutation with $F$ $\left( \frac{\sqrt{\text{Tr}(F^2 - FP)^2}}{\text{Tr}(F)} \right)$. mHartree accuracy in energy was confirmed by comparing the resulting energy with that of the diagonalization results in smaller molecules. In larger ones, the energies were compared among the linear scaling methods. To speed up SCF convergence $F$ was extrapolated using DIIS[68, 69].
Reliability

Within CGDMS $P$ is not specifically constrained to be idempotent, but is penalized implicitly for loss of idempotency. Thus, it is possible for runaway solutions to be found in which $P$ becomes non idempotent. In practice, this usually occurs when the starting $P$ guess is far from idempotency, when the neglect threshold is too loose and numerical instabilities appear, or when the system under study is non-physical (e.g., molecules with unphysically short bond lengths). As mentioned above, using a neglect threshold of $5 \times 10^{-5}$ a.u., we found that CGDMS reproduces the diagonalization result in polyglycines, water clusters and DNA fragments to mHartree accuracy in energy. However, when the neglect threshold is loosened too much above $5 \times 10^{-7}$ a.u., idempotency may be lost. For graphene sheets, which have a smaller HOMO–LUMO gap than the other systems under consideration, CGDMS must have a tighter neglect threshold of $1 \times 10^{-6}$ a.u. to maintain idempotency.

Idempotency is also an issue in CEM calculations. With some systems such as RNA molecules, the density matrix calculated using CEM is far from idempotency early on in the SCF iterations and only becomes idempotent later. Thus, several non-optimum steps are taken causing the calculation to require more SCF iterations to converge. This early lack of idempotency results from $F$ having a much smaller HOMO–LUMO gap during the first SCF cycle than in later cycles. For example, the HOMO–LUMO gap for a 368 atom DNA molecule [80] grows by a factor of 3.5 during the SCF iterations. This problem can be corrected by using a larger CPO for the early SCF iterations. However, in every system tested, CEM was
able to find the same energy as diagonalization even when the other methods could not. For example, calculations on C_{60} buckminsterfullerene using PD, CGDMS, and PDM converge to an excited state unless the virtual orbitals are level shifted several tenths of a Hartree. But even with this level shifting, PD, CGDMS, and PDM calculations require about 100 SCF iterations to converge to the ground state energy. However, CEM converges to the correct energy in just 10 SCF cycles.

Like CEM, PDM also has more difficulty with the first SCF iterations than later ones. During the first SCF iterations, it sometimes takes more than twice the number of purification cycles to obtain a certain degree of idempotency than in later iterations. This problem can be solved by shifting the virtual orbitals a few tenths of a Hartree higher in energy during the first few iterations. This is accomplished by adding $E_{\text{shift}}(I - P_{n-1})$ to $F$, where $E_{\text{shift}}$ is the amount of energy the virtual orbitals are to be shifted.

Idempotency is not an issue with PD; however, in some systems such as diamond chunks, PD using the initial guess from Lewis dot structures fails to find the correct solution even after hundreds of SCF iterations. This occurs because, as mentioned before, pseudodiagonalization is only a first order approximation to diagonalization. In most systems the elements of the $F_{\text{av}}$ block after the annihilation are nearer to zero than before. However, in some compact systems such as diamond chunks, the norm of the $F_{\text{av}}$ matrix actually grows after each annihilation for the first twenty or more SCF cycles. The norm never goes below the desired threshold unless PD is allowed to iterate for many hundreds of cycles. This problem is fixed by level shifting the virtual orbitals several tenths of a Hartree before the annihilation. CGDMS, on
the other hand, does not have this same problem with diamond chunks. A typical CGDMS calculation on diamond will converge in about 10 SCF iterations.

Performance

Unless otherwise stated, calculations were performed on a single MIPS R10K/195 MHz processor of a SGI Origin2000 computer. Figures 10 12 show the timing results of one entire SCF iteration using PD, CGDMS, PDM, and CEM for calculations on linear polyglycine chains[35], water clusters[36] and RNA molecules[76].

Linear scaling is achieved for a single SCF iteration using each of these methods in all of the systems studied. All of the methods are slower when used on RNA molecules than on water clusters and polyglycine molecules. This occurs because RNA molecules are more compact than the linear polyglycine chains and therefore produce more dense matrices. Water clusters, although compact, have only short-ranged interactions between molecules and thus also produce relatively sparse matrices. In all cases, CEM is about three to five times more expensive per SCF iteration than PD and CGDMS. This can be explained by the larger number of matrix multiplications and by the slightly smaller neglect threshold required by CEM. PD is slightly faster than CGDMS in every system but the polyglycine chains. CGDMS is competitive with PD even though it requires more matrix multiplies because CGDMS uses a “fixed” matrix form which is more efficient than the LIG style matrix multiplication used by PD. PDM is competitive with PD and CGDMS for polyglycine and RNA calculations. However, for water clusters calculations PDM is about two times more expensive than the other methods. This is
because fewer CGDMS and PD iterations are required for water clusters than the other systems, but PDM requires about the same number of purification cycles in all of the systems studied here. Thus, there is a reduction in computational effort for CGDMS and PD, but no such reduction is realized in PDM calculations.

Figures 13-15 show the number of SCF cycles required to converge the energy to mHartree accuracy. This number increases slightly with system size for all of the methods shown. However, if the energy per atom is converged to the same value during each calculation, the number of SCF cycles for a single point energy calculation will be constant with system size. The number of SCF iterations needed to converge the total energy to mHartree accuracy differs among the various methods for replacing diagonalization, and is not consistent among the various systems. PD calculations converge more quickly than CGDMS for polyglycine chains. However, for water clusters, the two methods reach convergence in the same number of iterations and for RNA molecules, CGDMS calculations converge slightly faster than PD. The number of SCF iterations required by CEM to reach convergence in polyglycine chain calculations oscillates around that of CGDMS. However, CEM calculations converge more quickly than CGDMS and PD for water clusters but more slowly than CGDMS for RNA molecules. The reason CEM converges slowly for RNA molecules is because $P$ is far from being idempotent during the early SCF iterations and only becomes idempotent later in the calculation. Thus, additional SCF cycles are needed to correct for the non-optimum steps taken while $P$ was non-idempotent. The larger molecules are more greatly affected by this phenomenon because more SCF cycles are needed before the density matrix becomes
idempotent. Like with CEM, the number of SCF iterations required for PDM also oscillates with system size. However, PDM usually converges more quickly than the other methods.

Table 5 shows CPU timings of single point energy calculations on several large molecules converged to mHartree accuracy. As shown, PD, PDM and CGDMS are very fast replacements for diagonalization, able to handle calculations on insulating systems with more than 10,000 atoms in reasonable amounts of CPU time.

Memory requirements for calculations on polyglycines are shown in Figure 16. The memory requirements for each method scale linearly with system size. In our implementation, PDM requires the least amount of memory of all the methods. CGDMS is more efficient than PD and CEM which both require about the same amount of memory. The reason CEM is competitive in memory with PD is because in CEM, only the diagonal elements of the CPO recursion matrices \( T_i \) need be stored so that \( \mu \) can be found without recalculating \( T_i \) every Newton Rapson step\[2\]. The bulk of the CEM memory requirements comes from storing three \( T_i \) matrices at one time while calculating the recursion matrices. However, since the columns of \( T_i \) can be calculated independently of one another the memory could be reduced to only enough to store several columns of the density matrix at one time.

Conclusions

In our implementation, CEM is a much slower method than PD, PDM and CGDMS for replacing diagonalization in large scale calculations. This conclusion differs from that of our tight-binding (TB) implementation of CEM[81. 2]. Unlike TB, the
semiempirical methods discussed in this paper require a self-consistent determination of the density matrix. For reasons discussed above, self-consistent calculations require many fewer CGDMS steps per SCF iteration than those required to form the TB density matrix. (Self consistent methods require between 2 to 4 CGDMS steps per SCF iteration while the TB method requires about 15.) No such reduction in CPU expense is obtained for CEM. Thus, although CEM is faster than CGDMS in the TB method, CGDMS is much faster than CEM in self consistent methods. However, since CEM usually converges to the correct energy when the other methods fail (e.g., calculations on C₆₀), it can be used as a backup for PD, PDM and CGDMS. Also, since CEM and PDM do not require an existing $P$ to update but only require $\mathbf{F}$ to form the new $\mathbf{P}$, these methods could be used to form a guess density matrix from the Hückel Hamiltonian as an alternative to the Lewis structure guess.

PD, PDM and CGDMS are comparable in CPU cost, but they fail under different circumstances. Thus, it is useful to have all methods available and implemented so that in cases where one method fails, the other can be used instead. However, I recommend CGDMS over the other methods because of its unique SCF convergence properties mentioned in Chapter 8. For some molecules, CGDMS has difficulty maintaining idempotency on the first SCF iteration of the calculation. This can be corrected by using PD for the first iteration and CGDMS afterwards. This allows the calculations to converge in fewer SCF cycles as demonstrated by the RNA molecule calculations in Figure 16. In conclusion, PD, CGDMS, PDM and CEM provide important alternatives to diagonalization in calculations on large
molecules. Not only is linear CPU scaling with system size attained, but also, these methods are parallelizable, which reduces CPU time significantly on parallel machines.
Table 5  Total CPU times for single point energy calculations on several large molecules using a single processor of a SGI Origin2000 machine. The neglect thresholds and convergence criteria are set to ensure nHartree accuracy in energy as discussed in the text. The column marked DG contains CPU time estimates for calculations using Householder diagonalization based on its $O(N^3)$ scaling properties and the average number of SCF iterations (in parentheses, from Figures 13–15) required by the other methods in the table. The entries marked with an * are estimated based on the linear scaling properties of CEM.

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Number of Atoms</th>
<th>CPU Time (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CGDMS</td>
<td>PD</td>
</tr>
<tr>
<td>2856–glycine</td>
<td>19,995</td>
<td>1.72</td>
</tr>
<tr>
<td>1700–glycine</td>
<td>11,903</td>
<td>0.96</td>
</tr>
<tr>
<td>4110–water</td>
<td>12,330</td>
<td>1.45</td>
</tr>
<tr>
<td>6304–RNA</td>
<td>6,304</td>
<td>4.72</td>
</tr>
<tr>
<td>3152–RNA</td>
<td>3,152</td>
<td>2.15</td>
</tr>
</tbody>
</table>
Figure 10  CPU time requirements for a single SCF iteration in polyglycine chain energy calculations using CEM, PD, PDM, and CGDMS. Neglect thresholds are set to ensure mHartree accuracy. A level shift of 0.2 Hartrees is added to $F$ in the PDM calculations and is reduced to 0 by the fifth SCF iteration using a linear function. A CPO of 75 is used for the CEM calculations. The number of CGDMS and PD iterations in each SCF cycle is 2 in both cases. 2 purification steps are applied each CGDMS iteration.[3]
Figure 11  CPU time requirements for a single SCF iteration in water cluster energy calculations using CEM, PD, PDM, and CGDMS. Neglect thresholds are set to ensure mHartree accuracy. A CPO of 60 is used for the CEM calculations. The number of CGDMS and PD iterations in each SCF cycle is 2 and 1, respectively. 2 purification steps are applied each CGDMS iteration.[3]
Figure 12 CPU time requirements for a single SCF iteration in RNA molecule energy calculations using CEM, PD, PDM, and CGDMS. Neglect thresholds are set to ensure mHartree accuracy. A PD step is performed during the first SCF iteration in the CGDMS calculations. A level shift of 0.2 Hartrees is added to $F$ in the PDM calculations and is reduced to 0 by the fifth SCF iteration using a linear function. A CPO of 100 is used for the CEM calculations. The number of CGDMS and PD iterations in each SCF cycle is 3 and 2, respectively. 2 purification steps are applied each CGDMS iteration.[3]
Figure 13 Number of SCF iterations required for single point energy calculations on polyglycine chains using CEX, PD, PDM, and CGDMS. Neglect thresholds are set to ensure mHartree accuracy. A level shift of 0.2 Hartrees is added to $F$ in the PDM calculations and is reduced to 0 by the fifth SCF iteration using a linear function. A CPO of 75 is used for the CEM calculations. The number of CGDMS and PD iterations in each SCF cycle is 2 in both cases. 2 purification steps are applied each CGDMS iteration.[3]
Figure 14  Number of SCF iterations required for single point energy calculations on water clusters using CEM, PD, PDM, and CGDMS. Neglect thresholds are set to ensure mHartree accuracy. A CPO of 60 is used for the CEM calculations. The number of CGDMS and PD iterations in each SCF cycle is 2 and 1, respectively. 2 purification steps are applied each CGDMS iteration.[3]
Figure 15  Number of SCF iterations required for single point energy calculations on RNA molecules using CEM, PD, PDM, and CGDMS. Neglect thresholds are set to ensure mHartree accuracy. A PD step is performed during the first SCF iteration in the CGDMS calculations. A level shift of 0.2 Hartrees is added to $F$ in the PDM calculations and is reduced to 0 by the fifth SCF iteration using a linear function. A CPO of 100 is used for the CEM calculations. The number of CGDMS and PD iterations in each SCF cycle is 3 and 2, respectively. 2 purification steps are applied each CGDMS iteration. [3]
Figure 16  Amount of memory required for polyglycine chain single point energy calculations using CEM, PD, PDM, and CGDMS. Neglect thresholds and CPO are set to ensure mHartree accuracy.[3]
Chapter 13

Conclusions

With the techniques mentioned in this work, I was able to reduce the time scaling of semiempirical methods from cubic to linear with the number of atoms in the chemical system. To the best of my knowledge, this work has provided the first adaptations of the conjugate gradient density matrix search (CGDMS), the chebyshev expansion method (CEM), and the purification of the density matrix (PDM) methods for self-consistent semiempirical methods. This made possible the largest semiempirical self-consistent geometry optimization on a protein using a diagonalization replacement other than divide and conquer. In addition, it has given a performance comparison of CGDMS, CEM, PDM and pseudodiagonalization (PD). CEM performs much more slowly than the other diagonalization replacements because it requires more matrix multiplications than the other methods. CGDMS, PDM and PD all require a similar amount of CPU time, but they all fail under different circumstances. Even though there are cases in which PDM and PD outperform CGDMS. I recommend CGDMS over the other methods because of its SCF convergence properties.

Future work could involve adapting CGDMS and PDM with a promising new set of purification transformations which can be used to obtain cubic convergence to an idempotent matrix. This is a significant improvement over the McWeeny
transformation which only converges matrices to idempotency quadratically\cite{82}. This adaptation would increase the efficiency of the purification transformations in these methods and would thus result in fewer matrix multiplications and a faster calculation.

Another possible future improvement is in the area of the initial guess density matrix. An alternative to the Lewis dot guess described in Chapter 3 could be to use PDM or CEM to form an initial guess density matrix from the Hückel Hamiltonian. The advantage of using these methods is that they do not require an initial guess $P$ or a set of molecular orbitals as PD and CGDMS do, so they are natural choices for forming a guess density matrix and could be used in cases where the Lewis dot guess is poor.

In addition, to reduce the error of semiempirical calculations, future research could involve an adaptation of the fast multiple method (FMM) to the semiempirical integrals which would replace the soft atom-atom distance cutoff criterion. Also, the parametrization of the current semiempirical methods was done using relatively small organic molecules. Future research needs be done in re-parametrizing the semiempirical methods for large biological molecules, as well as in determining the best uses for these computations in biological systems.

In conclusion, as computers become faster, linear scaling semiempirical and quantum chemical methods are becoming more important because of their potential use for biological systems. This work is a step toward making such large calculations possible.
References


[35] The polyglycine chains are similar to those described in Refs. [30, 31, 32]. Coordinates for these molecules are available from us.

[36] The water clusters up to 150 water molecules are those used in ref. [15]. We obtained the coordinates from the authors of ref. [15]. Coordinates for the larger water molecule clusters are available from us.


[47] During each SCF iteration between 2 to 4 CGDMS iterations are needed to produced a sufficiently converged density matrix. During each CGDMS step 5 matrix multiplications are required to form the gradient and 3 to calculate the step size. 2 more matrix multiplications are required to perform each purification transformation. To calculate the form of the matrix to be used in the CGDMS step, 2 matrix multiplies are needed and 2 matrix multiplies are needed for the DIIS step. Thus, a calculation with 2 CGDMS steps per SCF it-
eration and 2 purification cycles per CGDMS step requires a total of 28 matrix multiplies per SCF cycle.


[52] Ö. Farkas and H. B. Schlegel. "Methods for optimizing large molecules. II. Quadratic search." to be published in JCP.


[54] Ö. Farkas and H. B. Schlegel. "Methods for optimizing large molecules. III. A computationally economical and efficient optimization algorithm based on the GDIIS method." to be published in JCP.


[74] During each PD iteration 2 matrix multiplications are required to form $F_{\alpha\beta}$, and 1 more to rotate the orbitals. (The rotation is similar to a matrix multiplication in CPU cost.) Every SCF iteration 1 more matrix multiplication is required to form estimates for the eigenvalues of the matrix. 1 to form $P$, and 2 for DIIS. Hence, a calculation taking 2 PD steps per SCF cycle requires 10 matrix multiplications per SCF cycle.


[76] The 368 and 1576 atom RNA molecules are from Refs. [80] and [84], respectively. The 651 atom DNA molecule is from Ref. [83] with sodium atoms added to balance the charge. Hydrogens were added with Xplor. All larger RNA molecules were made from replicating the 1576 atom RNA molecule.

[77] CEM requires 2 less than CPO matrix multiplications to form $T_i$. In addition it requires 2 matrix multiplications to determine the form of the matrices, and 2 for DIIS.

[78] PDM requires 2 matrix multiplications per purification cycle. It also requires 2 matrix multiplications to determine the form of the matrices, and 2 for
DIIS. So a PDM step using 9 purification transformations requires 22 matrix multiplications per SCF cycle.


