Sparse Matrices in Self–Consistent Field Methods

Emanuel Rubensson

Theoretical Chemistry
Royal Institute of Technology
Stockholm 2006
Sparse Matrices in Self–Consistent Field Methods
Licentiate thesis
© Emanuel Rubensson, 2006
ISBN 91-7178-534-5
Printed by Universitetsservice US AB,
Stockholm, Sweden, 2006
Typeset in \LaTeX by the author.
Abstract

This thesis is part of an effort to enable large-scale Hartree–Fock/Kohn–Sham (HF/KS) calculations. The objective is to model molecules and materials containing thousands of atoms at the quantum mechanical level. HF/KS calculations are usually performed with the Self-Consistent Field (SCF) method. This method involves two computationally intensive steps. These steps are the construction of the Fock/Kohn–Sham potential matrix from a given electron density and the subsequent update of the electron density usually represented by the so-called density matrix. In this thesis the focus lies on the representation of potentials and electron density and on the density matrix construction step in the SCF method. Traditionally a diagonalization has been used for the construction of the density matrix. This diagonalization method is, however, not appropriate for large systems since the time complexity for this operation is $O(n^3)$. Three types of alternative methods are described in this thesis; energy minimization, Chebyshev expansion, and density matrix purification. The efficiency of these methods relies on fast matrix–matrix multiplication. Since the occurring matrices become sparse when the separation between atoms exceeds some value, the matrix–matrix multiplication can be performed with complexity $O(n)$.

A hierarchic sparse matrix data structure is proposed for the storage and manipulation of matrices. This data structure allows for easy development and implementation of algebraic matrix operations, particularly needed for the density matrix construction, but also for other parts of the SCF calculation. The thesis addresses also truncation of small elements to enforce sparsity, permutation and blocking of matrices, and furthermore calculation of the HOMO–LUMO gap and a few surrounding eigenpairs when density matrix purification is used instead of the traditional diagonalization method.
Preface

Comment on my contribution to the included papers  The included papers are listed and briefly summarized in chapter 5. I was responsible for the development and implemention of the data structure and methods presented in papers 1–4. I was also, to a great deal, responsible for preparation and editing of manuscripts for these papers. In paper 5 the work was equally distributed between the authors.
Acknowledgment

A special thanks to my eminent supervisor Dr. Pawel Salek for his guidance and support. Thanks also to Prof. Hans Ågren for his belief in my academic capabilities and for giving me the opportunity to study at the Department of Theoretical Chemistry. Part of the work presented in this thesis was carried out at the University of Southern Denmark (SDU), Odense, under the supervision of Prof. Hans Jørgen Aagaard Jensen. I am very grateful for the hospitality that has been shown to me by Hans Jørgen, Hans Aage and Jette Nielsen, and Dr. Emmanuel Fromager during my stays in Odense. Thanks to Elias Rudberg for a very fruitful collaboration. And, to all the people at the theoretical chemistry departments at KTH and SDU for contributing to two nice working places. Thanks also to Paweł, Hans, Hans Jørgen, Emmanuel, Emil Jansson, Peter Hammar and Sara Zahedi for valuable comments during the preparation of this thesis. This research has been supported by the Sixth Framework Programme Marie Curie Research Training Network under contract number MRTN-CT-2003-506842 and the NorFA network in natural sciences “Quantum Modeling of Molecular Materials”. Finally, I would like to thank my family, my friends and my beloved Sara for their love and support.
## Contents

Abstract iii  
Preface v  
Acknowledgment vii

### I Introductory Chapters 1

1 Introduction 3  
  1.1 Outline of thesis 4

2 The Self–Consistent Field Method 7

3 Density Matrix Construction 11  
  3.1 Three types of density matrix algorithms 11  
  3.2 Convergence criteria for the purification process 17

4 Sparse Matrix Representation 19  
  4.1 Sparsity and nonzero patterns 19  
  4.2 Matrix data structures 21  
  4.3 Performance 23

5 Included Papers 25

6 Conclusion 27
A Supporting Information

A.1 Definitions ................................................................. 29
A.2 Implementation ............................................................ 30
   A.2.1 Proxy objects .......................................................... 30
   A.2.2 Matrix templates ....................................................... 31
   A.2.3 Matrix–matrix multiplication ....................................... 32
   A.2.4 Supported operations in the Hierarchic Matrix Library ....... 33

Bibliography ........................................................................ 35

II Included Papers ................................................................ 39

1 Systematic sparse matrix error control for linear scaling
electronic structure calculations ........................................ 41

2 Determination of the chemical potential and HOMO/LUMO orbitals in
density purification methods ............................................. 53

3 A hierarchic sparse matrix data structure for large–scale Hartree–Fock/Kohn–
Sham calculations .............................................................. 65

4 Sparse matrix algebra for quantum modeling of large systems .... 81

5 Near–idempotent matrices ................................................... 93
PART I

Introductory Chapters
Chapter 1

Introduction

During the twentieth century, theory and methods that made it possible to model materials and molecules at the electronic level emerged. A milestone in this progress was the development of the Hartree–Fock (HF) method in the twenties to approximately model the electronic Schrödinger equation[1]. In the HF theory, electron movement is simulated in the average field of all other electrons and nuclei in the system. The most critical problem with the HF theory is that electron correlation is neglected. One way to deal with this problem is via so-called post–HF methods in which the HF wave function is used as a starting point for more accurate calculations. These methods, however, involve a considerably increased computational cost. In the sixties, another approach called density functional theory (DFT) appeared, which in principle can solve the electron correlation problem[2]. In the original version of DFT, the total energy is given by a functional of the electron density. However, in the most common version of DFT, proposed by Kohn and Sham (KS), only a part of the total energy, namely the exchange–correlation energy, is given by the functional[3] and the remainder of the energy is treated in the same way as in the HF method. From a computational point of view, the KS method is similar to the HF method. The HF and KS theories both result in an eigenvalue equation that is almost universally solved with the Self–Consistent Field (SCF) method, outlined in chapter 2. It is therefore often unnecessary to distinguish between the HF method and the KS method, especially in the matters treated in this thesis, thus I will in the following refer to them together as the HF/KS method whenever possible.

Twenty years ago, the number of electrons possible to simulate with the HF/KS method was limited by computer performance. After a tremendous increase in computer capacity, the limiting factor moved to the algorithmic level. The reason is that the algorithms used in early implementations of the HF/KS method have a computational time complexity of \( O(n^4) \) where \( n \) is the system size. In other words, increasing the system size by a factor
of 2 led to a 16 times longer execution time. The memory usage situation was better with a memory complexity of $O(n^2)$. In recent years much research has been devoted to the development of new algorithms to improve the scaling of the HF/KS method. For large systems the computational time is dominated by two steps in the SCF calculation. The first step is the construction of the Fock/Kohn–Sham matrix for a given electron density. For this step, algorithms like the well known fast multipole method has been developed[4, 5, 6, 7, 8]. The second step is the update of the electron density for a given Fock/Kohn–Sham matrix. For this step a number of methods have been presented, for example different variants of density matrix purification[9, 10, 11, 12, 13].

Problems will appear as larger systems become available for investigation. While some of the problems cannot yet be seen, others can be predicted. For example, many functionals that exist today are designed for rather small systems. Potential surfaces with many local minima will most likely be a more pronounced problem for large systems. The SCF convergence may behave differently for large systems leading to a need for well–defined convergence supervision/acceleration methods. Solutions to this kind of problems cannot, however, be tested until larger systems become available for study. Therefore, the practical problems with memory and time usage need to be solved first.

This thesis concerns an efficient representation of electron density and potentials in large–scale HF/KS computations. This representation is intended to reduce the overall memory usage and the time usage in matrix operations, particularly needed for the density matrix construction, but also in other parts of the SCF calculation. For the density matrix construction, density matrix purification has been used in this thesis, and some material is presented with respect to this. The practical problems that regard construction of Fock and Kohn–Sham matrices from given trial electron densities are not treated in this thesis. For Fock matrix construction I refer to the thesis by my colleague, Elias Rudberg[14].

1.1 Outline of thesis

The introductory chapters are intended to make the reading of the following papers more rewarding. Chapter 2 presents the computational problem that is the result of the HF/KS theories. One part of the computational problem regards the density matrix construction. Chapter 3 gives an overview of different approaches to construct the density matrix. All these approaches rely on an efficient representation of matrices and fast matrix–matrix multiplication. In chapter 4 different data structures and treatment of matrices are discussed. In this chapter I have also included some benchmarks of Hartree–Fock calculations on DNA double strands of increasing system size. In chapter 5 the main contributions of the included
papers are listed. The introductory chapters end with conclusions.
Chapter 2

The Self–Consistent Field Method

This chapter is intended to explain the computational problem that comes out of the Hartree–Fock (HF) and Kohn–Sham (KS) theories. Some concepts that are commonly used within these theories will here be used without definitions. In such cases, however, it is because they are not central to this thesis. For a more thorough survey of the HF and KS theories, see for example refs. [1, 3, 15, 16]. This chapter is outlined in the following way; First the computational problem is formulated. Next, the most common method to solve this problem is described. Finally, I discuss applications of this method to large systems.

The HF and KS theories both result in the eigenvalue equation

\[ FC = SCA \]  \hspace{1cm} (2.1)

Here, the Fock/Kohn–Sham matrix $F$ is the total potential matrix which is the sum of two parts; the one–electron matrix $H_1$ and the two–electron matrix $F_{2el}$. The $C$ matrix with eigenvectors is a coefficient matrix that expands the electron orbitals in a basis set

\[ \Phi(r) = C^T x(r) \]  \hspace{1cm} (2.2)

where $\Phi$ is a vector with orbitals and $x$ is a vector with basis functions, often built up from linear combinations of Gaussian functions $e^{-\alpha r^2}$ centered at the nuclei of the molecule. The diagonal eigenvalue matrix $\Lambda$ gives the energy for each orbital in $\Phi$, and $S$ is the basis set overlap matrix.

The $H_1$ matrix gives the electron kinetic energy and Coulomb attraction to the nuclei. This matrix is completely defined by the basis set and the positions and charges of the nuclei. The two–electron matrix $F_{2el}$, consists of two parts; Coulomb repulsion and correlation between electrons. While the Coulomb repulsion, given by the Coulomb matrix $J$, is the same in the HF and KS methods, the two methods differ in the other term; the exchange
correlation contribution. In the HF method only the exchange effects that origin from the anti-symmetrization of the wavefunction are included. These effects are contained in the so-called exchange matrix $K$. In the KS method, on the other hand, the electron correlation is given by a functional of the electron density. There exist also hybrid HF/KS methods where the two-electron matrix includes a mixture of HF and KS exchange correlation. The important thing to note here, however, is that the $F_{2el}$ matrix not only depends on the basis set and the number of electrons but also on the $C$ matrix. Expressions for the matrices introduced here can be found in section A.1.

The objective of HF/KS computations is to solve the eigenvalue equation 2.1. Since the Fock/Kohn–Sham matrix $F$ depends on its own eigenvectors $C$ a solution must satisfy an additional self-consistency condition; when the vectors used to build the matrix $F$ are also its eigenvectors, the equation is considered to be self-consistent. Equation 2.1 is almost universally solved with the SCF method which, in its simplest form, is a fixed point iteration. From now on, I will refer to the Fock/Kohn–Sham matrix as the Fock matrix.

**A fixed point iteration** Since the Fock matrix depends explicitly on some of its eigenvectors, equation 2.1 must be solved iteratively. If the set of eigenvectors that constitutes the Fock matrix are also its eigenvectors we say that we have a self-consistent field and a solution to the equation. The simplest way to find a self consistent field is via a fixed point iteration started by some initial guess, for example the solution given by neglecting two-electron interactions, i.e. the set of eigenvectors $C$ that solves the explicit eigenvalue equation $H_1 C = SCA$. The fixed point iteration involves two computationally demanding steps; 1) The construction of the Fock matrix from a given electron density. 2) The subsequent construction of the electron density for the computed Fock matrix. This procedure is referred to as the SCF cycle. The electron density $\rho(r)$ is represented by the eigenvectors or by the so-called density matrix $P$

$$\rho(r) = x^T(r)C_{occ}C_{occ}^T x(r) = x^T(r)Px(r) \quad (2.3)$$

where the $C_{occ}$ matrix consists of the eigenvectors that are associated with the $n_{occ}$ smallest eigenvalues, $n_{occ}$ is the number of occupied orbitals, and $P = C_{occ}C_{occ}^T$.

Unfortunately, the fixed point generally has a rather small stable region. This means that the fixed point iteration converges only if a good starting guess is provided. There has been many suggestions on how to improve the convergence of the simple fixed point iteration. A nice review of such improvements is given in ref. [17].

**SCF for large systems** The matrices $S$, $F$, and $P$ become sparse when the separation between atoms exceeds some system dependent value. The eigenvectors $C$, on the other
hand, are generally not sparse. Therefore, it is an advantage to use the density matrix as the primary variable rather than the eigenvectors. In the SCF cycle, the density matrix can be constructed directly from the Fock matrix, in a time that grows linearly with system size, without the intermediate calculation of the eigenvectors. In chapter 3, three different classes of density matrix constructed methods are described. The Fock matrix can be constructed from the density matrix without explicitly appearing eigenvectors, see section A.1. Methods also exist to make the Fock matrix construction time scale linearly with system size. The one–electron matrix and the Coulomb matrix can be constructed by use of the fast multipole method [4, 5, 6, 7, 8]. The Hartree–Fock exchange can also be calculated with linear complexity [18]. Also for the DFT exchange correlation term exists linear scaling methods, see for example ref. [19]. See ref. [14] for an overview and additional references to articles about Fock matrix construction.
Density Matrix Construction

The density matrix $P$ is usually constructed from the eigenvectors $C_{\text{occ}}$, $P = C_{\text{occ}}C_{\text{occ}}^T$. These eigenvectors are obtained from the solution of the eigenvalue equation 2.1. Since the time needed to perform this operation grows cubically with system size and the eigenvectors generally are not sparse, only systems with limited matrix size can be treated in this way. Therefore, alternative methods have arisen that tries to construct the density matrix directly from the Fock matrix. One can define the density matrix in terms of the Fock matrix in several different ways leading to exactly the same matrix. In this chapter three different possibilities are outlined. Termination criteria for density matrix purification is also discussed.

3.1 Three types of density matrix algorithms

First, we will need a few definitions. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n-1} \leq \lambda_n$ be the eigenvalues of the $n \times n$ Fock matrix $F$. The eigenvalues $\lambda_{n_{\text{occ}}}$ and $\lambda_{n_{\text{occ}}+1}$ are usually referred to as the HOMO and LUMO eigenvalues respectively. Here $n_{\text{occ}}$ is the number of occupied orbitals, HOMO is the highest occupied molecular orbital, and LUMO is the lowest unoccupied molecular orbital. The HOMO–LUMO gap is simply the interval $[\lambda_{n_{\text{occ}}}, \lambda_{n_{\text{occ}}+1}]$. Any value $\mu$ that lies in the HOMO–LUMO gap will, in this thesis, be referred to as a chemical potential. While some of the methods described here can also be used directly in a non-orthogonal basis, i.e. with $S \neq I$, it will be convenient to go over to orthogonal basis, that is, assume that $S = I$. This can for example be achieved by means of an inverse Cholesky decomposition of the overlap matrix $S^{-1}$. 

11
Energy minimization  The density matrix can be seen as a quantity that minimizes a certain energy. Here one needs to distinguish between the energy expression that is minimized when equation 2.1 is solved for a fixed Fock matrix and the one that is minimized when the additional condition of self-consistency is added. While it is possible to go for a minimization of the latter energy, this is more related to acceleration of the SCF convergence. The focus here is rather on methods that tries to obtain exactly the same result as from a diagonalization of the Fock matrix. The energy expression for a fixed Fock matrix is

\[ E = \text{Tr}[PF] \]  (3.1)

This energy expression is the starting point for energy minimization methods[20, 21, 22, 23, 24]. A starting guess is chosen and the energy is minimized by use of the steepest descent or the conjugate gradient method. The most important difference between methods of this class lies in how they handle the additional idempotency conditions

\[ P = P^2 \]  (3.2)
\[ \text{Tr}[P] = n_{\text{occ}} \]  (3.3)

Based on how the minimization methods treat the idempotency conditions, they can be divided into two classes. In the first class of methods, the density matrix in the energy expression (3.1) is replaced with its McWeeny-purified version \( 3P^2 - 2P^3 \) to penalize non-idempotent matrices[20, 21, 22]. A problem with this approach is that the searched minimum is not a global minimum since the eigenvalues of the trial density matrix \( P \) need to lie in the \([-0.5, 1.5]\) interval for the McWeeny purification to have effect. The trace criteria can for example be fulfilled by shifting the Fock matrix with the chemical potential[21]. This results in the following potential

\[ \Omega(P) = \text{Tr}[(3P^2 - 2P^3)(F - \mu I)] \]  (3.4)

In other methods, the search for the minimum is constrained to the idempotent manifold[23, 24]. In figure 3.1, the energy potential on the idempotent manifold is illustrated with a contour plot.

Chebyshev expansion  Another possible way to express the density matrix is via the Heaviside step function

\[ P = \theta(\mu I - F) \]  (3.5)

A truncated Chebyshev expansion could, in principle, be used to directly express this step function (after a scale and shift of the eigenvalues into the \([-1, 1]\) interval). The difficulty of expressing the discontinuity at the chemical potential, however, results in errors that
Figure 3.1: In one class of energy minimization methods, the search for the density matrix is carried out on the idempotent manifold. For a $3 \times 3$ system with $n_{occ} = 1$, this manifold is determined by the first two components $x_1$ and $x_2$ of the single eigenvector that constitutes the range space of the trial density matrix $P$. The third component is simply $x_3 = \sqrt{1 - x_1^2 - x_2^2}$, (the sign of the eigenvector is determined by the third component). The three eigenvectors of the Fock matrix $F$ are indicated in the figure with crosses. The eigenvalues corresponding to these eigenvectors have been chosen to $-1$, $0$, and $1$. The contour lines shows the energy $\text{Tr}[PF]$ for different choices of trial density matrices on the idempotent manifold. If the occupation count is changed to $n_{occ} = 2$, the second eigenvector must be chosen in the subspace orthogonal to the first eigenvector. The dashed line indicates this subspace for this example.
spread over the entire interval. This is problematic since we require the eigenvalues of the density matrix to be very close to zero or one. One way to improve this behavior is to let the Chebyshev expansion approximate a function that varies smoothly from 0 to 1 in the HOMO–LUMO gap, simply because this interval is free from eigenvalues. The most common choice is to use the so-called Fermi–Dirac function\[25, 26, 27, 28\]

\[
\frac{1}{1 + e^{-\beta(\mu-F)}}
\]

(3.6)

where $\beta$ depends on the HOMO–LUMO gap[28]. The smaller the HOMO–LUMO gap is, the larger $\beta$ is needed. In this way the oscillations are reduced, see Figure 3.2. However, the convergence towards 0 and 1 is anyway slow and therefore quadratically convergent purification is needed to refine the result, see the following paragraph. Another potential drawback with this method is that many intermediates are required for different powers of the scaled and shifted Fock matrix. Also, bounds for the HOMO and LUMO eigenvalues are needed to accurately choose $\beta$– and $\mu$–values. When the location of the HOMO and LUMO eigenvalues are not known, they are usually found by a trial and error approach where several density matrices are computed until the chemical potential is found[26]. While this is not as expensive as it seems, since powers of the scaled and shifted matrix can be reused, it requires more intermediate matrices to be stored. If the chemical potential is known, the HOMO and LUMO eigenvalues can instead be found by the spectral transformation Lanczos algorithm[28, 29].

**Density matrix purification** The density matrix $P$ is a projection matrix. The range space of $P$ is spanned by the eigenvectors of the Fock matrix $F$ that correspond to the $n_{\text{occ}}$ smallest eigenvalues. In the same way the null space of $P$ is spanned by the eigenvectors of $F$ that correspond to the $n - n_{\text{occ}}$ largest eigenvalues.

In density matrix purification methods, the starting guess has the correct eigenspace but wrong eigenvalues. An iterative process is used to push the $n_{\text{occ}}$ smallest eigenvalues towards 1 and the rest towards 0. A general purification algorithm, as well as three distinguished purification methods are described in paper 2. The grandfather of density matrix purification algorithms, the McWeeny purification, was developed already in the fifties[30]. The purpose of this algorithm was initially to refine a roughly idempotent matrix. That is, a matrix where the eigenvalues lie close to their desired values 0 and 1. More than forty years later the use of this algorithm was extended by Palser and Manolopoulos to replace the traditional diagonalization method[9]. A more useful algorithm that does not require prior knowledge of the chemical potential was also presented, and after that a number of purification algorithms appeared in the literature[10, 11, 12, 13]. A few examples of purification polynomials are given in figure 3.3.
One potential problem with the three alternatives to diagonalization presented here is that only the density matrix is obtained, and not the eigenvalues and eigenvectors. In paper 2, it is explained how the chemical potential can be obtained as a by–product of purification. Once the chemical potential is known, eigenpairs around the HOMO–LUMO gap can be found by the Spectral Transformation Lanczos method[29].

Relations between the density matrix algorithms  The classification of density matrix construction algorithms presented above is somewhat ambiguous. This is because some of the density matrix properties can be used to motivate and explain also other algorithms. The Chebyshev expansion and purification methods are clearly similar, since in both methods, a Heaviside step function is approximated by a polynomial expansion. In the following, I will sometimes refer to them together as polynomial expansion methods.

In the idempotency–preserving energy minimization algorithms the idempotency condition is fulfilled already in the starting guess, whereas the eigenvectors are adjusted in the minimization process. In the polynomial expansion methods, the eigenvectors are correct from the beginning and the eigenvalues are adjusted to their desired values. One could imagine a third dimension in figure 3.1 in which the eigenvalues vary. In the polynomial expansion methods, the only change during the iterations takes place in this third dimension, while in idempotent minimization, on the contrary, the only change occurs within the plane.
The McWeeny polynomial
The trace correcting purification polynomials
The canonical purification polynomials

Figure 3.3: Polynomials used in different purification methods to push eigenvalues towards 0 and 1. Note that all polynomials have fixed points at 0 and 1. In the McWeeny purification, eigenvalues are assumed to lie close to their desired values, 0 or 1, (the HOMO–LUMO gap must at least cover 0.5) and the polynomial pushes eigenvalues towards 0 and 1 simultaneously. In the other two methods one uses a polynomial sequence where the polynomials are chosen so that the resulting matrix fulfills the idempotency conditions without prior knowledge about the location of the HOMO–LUMO gap. The McWeeny polynomial is $3x^2 - 2x^3[30]$, the trace correcting purification polynomials are $x^2$ and $2x - x^2[10]$ and the canonical purification polynomial is $\frac{(1-2c)x + (1+c)x^2 - x^3}{1-c}$ if $c \leq \frac{1}{2}$ and $\frac{(1+c)x^2 - x^3}{c}$ if $c \geq \frac{1}{2}$ where $c$ is chosen to conserve the trace of the matrix[9]. In the figure, the canonical purification polynomial is plotted for $c = 0$ and $c = 1$. 

---

**Chapter 3**
shown in the figure. In the other class of minimization methods, the search is carried out in all dimensions.

The energy expression has also been used as a motivation for purification methods. See for example ref. [9] where the authors argue that the purification process constitutes a downhill energy minimization process and an energy expression is used in the termination criteria of the iterative process.

3.2 Convergence criteria for the purification process

Different suggestions for convergence criteria of density matrix purification exist in the literature. Palser and Manolopoulos use an energy expression to decide when to terminate the canonical purification process in ref. [9]. The purification is stopped as soon as the energy increases. Another suggestion is to stop the purification when \( \| X_{n+1} - X_n \| < \varepsilon \) where \( X_n \) is the current matrix in the purification process[13]. In paper 5, two theorems are presented that can be used in the convergence criterion. The idea is that the purification process can be stopped when each eigenvalue is sufficiently close to its desired value of 0 or 1. Since, in practice, eigenvalues are perturbed because of truncation, one cannot expect the eigenvalues to come closer to 0 and 1 than the magnitude of this perturbation. With the help of Weyl’s theorem[31] it can be seen that if the truncation is done so that \( \| X - \tilde{X} \|_2 < \tau \), where \( \tilde{X} \) is the truncated version of \( X \), then \( |\lambda_i - \tilde{\lambda}_i| < \tau \) where \( \lambda_i \) and \( \tilde{\lambda}_i \) are the eigenvalues of \( X \) and \( \tilde{X} \) respectively. Therefore, it is reasonable to stop the purification process as soon as each eigenvalue is within \( \sim \tau \) from its desired value. To find out when this is fulfilled one can use the theorems presented in paper 5. The criterion, that each eigenvalue should be close to its desired value, is in the paper divided into two parts; 1) Each eigenvalue should be close to 0 or 1. This criterion is in the paper related to \( \| X - X^2 \|_2 \). 2) The number of eigenvalues that are close to 0 and 1 respectively should be correct. This criterion is in the paper related to \( \text{Tr}(X) \) and \( \text{Tr}(X^2) \). It is assumed here that eigenvalues are kept ordered in the sense that each eigenvalue that should be 1 at convergence is always larger than each eigenvalue that should be 0 at convergence. It should be noted that, even though the polynomials used in the purification are monotonically increasing in the \([0, 1]\) interval, truncation could in principle cause eigenvalues to switch order.
This chapter gives an overview of the use of sparse matrices in the HF/KS method. The chapter starts with a discussion about sparsity and nonzero patterns and how they can be utilized. Next, alternatives for the representation is described and it is explained why the hierarchic data structure, presented in paper 3 and 4, was developed. The chapter ends with benchmarks of density matrix constructions using trace correcting purification implemented in the Hierarchic Matrix Library.

4.1 Sparsity and nonzero patterns

Each element in the Fock matrix $F$, the overlap matrix $S$ and the density matrix $P$ corresponds to two basis function centers that are usually located at the atom centers in the molecule, see figure 4.1. The magnitude of the matrix elements generally depends on the distance between the two atom centers, larger distances gives smaller elements. If small matrix elements are removed, we can take advantage of the sparsity that appears for molecular systems reaching out sufficiently far in space. That is, one can avoid to store and address zero matrix elements. Furthermore, since the molecular geometry, e.g. the three–dimensional arrangement of the atoms, is known, the dependency between magnitude and distance also gives information about nonzero patterns in the matrices. Thus, matrix elements can be treated in groups, which reduces the addressing overhead and gives more efficient matrix algebra, since optimized libraries, such as the Basic Linear Algebra Subprograms (BLAS)[32], can be used for submatrix operations.

**Truncation to enforce sparsity** Several methods exist to enforce matrix sparsity. I will describe here three methods; In these methods the truncation is based on a cutoff radius,
Figure 4.1: Each matrix element corresponds to two atom centers. Hence, each element is associated with a distance between two atoms.

the magnitude of matrix elements, or on the norm of the matrix.

In the cutoff radius based methods, the decay of element magnitude with distance is directly utilized[9, 21, 23, 24, 25, 27, 28, 33]. Each matrix element corresponds to two atoms. If elements are grouped into submatrices, each submatrix corresponds to two groups of atoms. If the shortest distance between the two groups is greater than a predefined cutoff radius the submatrix is dropped. There are a few problems with this approach. Usually, the exact correspondence between cutoff radius and matrix element magnitude is not known. Even though it has been shown that the magnitude decay rate is related to the HOMO–LUMO gap of the system[34], rigorous error control seems difficult.

Another approach to remove small elements is to drop all matrix elements below a predefined threshold[20, 34, 35]. If elements are grouped into submatrices, submatrices are dropped when their norms are below the threshold[11, 22]. In this way a rigorous error control is possible. However, if one desires to relate the truncation threshold to properties like error in energy, accuracy in eigenvalues, etc one has to assume that the errors in all elements/submatrices due to truncation are equally large. This assumption becomes increasingly pessimistic as the matrix size increases.

A third possibility, presented in paper 1, is to carry out the truncation based on the norm of the entire error matrix $E$. If $A$ is the matrix before truncation, $A + E$ is the truncated matrix. The truncation is performed so that

$$\| E \| < \tau$$  \hspace{1cm} (4.1)

where $\tau$ is a threshold possibly depending on the desired accuracy of some property of the system, for example the total energy. The choice of norm is important. If the norm is defined as the maximum single element/submatrix error, this formulation reduces simply to the element based truncation. With the Euclidean norm one can use important theorems such as Weyl’s theorem about the maximum movements of eigenvalues for a certain threshold,
Sparse Matrix Representation

see for example section 3.2. However, truncation based on the Frobenius norm is easier to implement and theorems based on the Euclidean norm can still be used since the Frobenius norm is an upper bound to the Euclidean norm. Sometimes, however, a bound is not sufficient and the bound generally gets poorer as the system size increases. Therefore, it will be worthwhile to implement also some kind of truncation based on the Euclidean norm.

Permutation to exploit nonzero patterns The goal of the permutation is that zero matrix elements end up in some submatrices and nonzero elements in others. The matrix elements can then be referred to collectively via the submatrices in the matrix data structure. As a first approach, Challacombe proposed to group basis functions belonging to the same atom[22]. Later on, it was recognized that the performance in submatrix operations could be improved if the basis functions belonging to several atoms are grouped into larger multi–atom blocks[33]. This also exposed a trade–off: On one hand, it is good to have large blocks, to reach higher speed in submatrix operations. On the other hand, small blocks are good, since more zero elements can be squeezed out of the data structure than with large blocks.

While atom– and multi–atom blocks may be appealing from a physical point of view, this type of blocking is often suboptimal since the block size can not be chosen at will. This is problematic since small changes in block size often gives large performance drops in the underlying BLAS routines. Also, atom– and multi–atom blocks give nonuniform block sizes since different types of atoms have different number of basis functions centered on them. This gives an increased probability of heap memory fragmentation. For these two reasons a uniform block size has been used in papers 3 and 4. This block size can be tuned with respect to hardware and BLAS libraries.

4.2 Matrix data structures

In previous implementations of sparse matrix algebra for large–scale HF/KS calculations, the Blocked Compressed Sparse Column (BCSC) representation was used. In this section the BCSC data structure will be described starting from the original unblocked version, the Compressed Sparse Column (CSC) representation. The objective is to explain why the BCSC data structure was preferred in front of the CSC data structure and what difficulties with the CSC and BCSC data structures that are avoided in the hierarchic data structure.

Compressed Sparse Column format The most common sparse matrix data structure is the CSC format[36, 37], used for example in the technical computing program Matlab[38, 39]. The CSC representation of a matrix is given by three one–dimensional arrays. Two arrays
list the row indices and the numerical values of all nonzero matrix entries (in column-wise order). The third array list column pointers where the $i$:th element is the index, in the two other arrays, of the first nonzero element of the $i$:th column of the matrix. For example, consider the matrix

$$A = \begin{pmatrix} 7 & -3 \\ 8 & 5 \\ -2 & 6 \end{pmatrix}$$

A CSC representation of $A$ is given by

- column pointers = 0 1 3 5 7
- row indices = 0 1 3 0 2 1 3
- numerical values = 7 8 $-2$ $-3$ 1 5 6

This representation takes advantage of sparsity but does not care about data locality. Another problem is that matrix algorithms that operate on this data structure are comparatively complicated to develop and implement. As an example, see the CSC matrix–matrix multiplication described in section A.2.3.

**Blocked Compressed Sparse Column format** Since *a priori* information about the locations of nonzero matrix elements exist, it is possible to reorder the matrix so that nonzeros end up in some submatrices and zeros in others. With our example matrix $A$ this can be done by switching the second and the third row and column and block the matrix into four $2 \times 2$-submatrices. We can then use the CSC format but let the indices address blocks rather than single matrix elements. A BCSC representation of the reordered matrix is given by

- column pointers = 0 1 2
- row indices = 0 1
- submatrices = \[
\begin{pmatrix} 7 & -3 \\ 1 \end{pmatrix} \begin{pmatrix} 8 & 5 \\ -2 & 6 \end{pmatrix}
\]

This gives a data structure that takes advantage of both sparsity and nonzero patterns. Nevertheless, the algorithmic problem inherited from the CSR representation remains.

**Hierarchic data structure** Fortunately, computer science has made progress not only in hardware, but also in development of programming languages and compilers. Using modern techniques, it is therefore possible to develop simplified data structures. The hierarchic
Sparse Matrix Representation

4.2 Performance

Figure 4.2: DNA double strands \((A - T)_n\), \(n = 4, 8, 16, 32\) used for benchmarks to represent typical molecules of relevance in current chemical research. These same kind of systems with \(n = 1, 2, 4, 8, 16\) were used in ref. [18] for benchmarks of Fock matrix construction.

data structure, presented in papers 3 and 4, uses generic programming to obtain a high-performing and transparent code. Symmetry has been utilized, which reduces CPU time and memory usage by more than 45% in symmetric operations. Previously, reduction of CPU time by 30% for sparse symmetric operations in electronic structure calculations has been reported[28]. For a description of the hierarchic data structure, I refer to papers 3 and 4. See also section A.2 for an introduction to some techniques used in the implementation.

4.3 Performance

Figure 4.3 shows timings for trace correcting purification, applied to the DNA double strands pictured in figure 4.2, compared to diagonalization with the Math Kernel Library on an Intel Xeon EM64T 3.4 GHz. The trace correcting purification was implemented with use of the hierarchic data structure and symmetric matrix squares.
Figure 4.3: Benchmarks for Hartree–Fock computations on DNA double strands (see figure 4.2) with basis set 3–21G, truncation threshold $\tau = 10^{-3}$ in the Frobenius norm. Panel (a) shows the time used for trace correcting purification compared to diagonalization. The diagonalization time was estimated from a reference calculation for a matrix size of ca. 8000. Panel (b) shows how the matrix sparsity changes with increased systems size, for the density matrix, the Fock matrix, and the overlap matrix.
Chapter 5

Included Papers

Paper 1: Systematic sparse matrix error control for linear scaling electronic structure calculations
In the first paper the main contribution is efficient truncation criteria for blocked sparse matrix data structures. The novelty lies in that the truncation is based on the norm of the entire error matrix rather than the errors in single matrix elements or sub-matrices. This paper discusses also accumulated errors in the trace correcting purification algorithm.

Paper 2: Determination of the chemical potential and HOMO/LUMO orbitals in density purification methods
In this paper it is shown how the chemical potential can be obtained as a by-product of density matrix purification. Eigenpairs around the HOMO–LUMO gap are calculated with the Spectral Transformation Lanczos method.

Paper 3: A hierarchic sparse matrix data structure for large–scale Hartree–Fock/Kohn-Sham calculations
The hierarchic matrix data structure is presented in this paper. The data structure allows for easier implementation of matrix algebra. Matrix symmetry is utilized and increases the computational speed and reduces memory usage by a factor close to 2.

Paper 4: Sparse matrix algebra for quantum modeling of large systems
This paper summarizes the results of papers 1 and 3. Also, new benchmarks are presented and it is shown that the truncation method proposed in paper 1 leads to predictable errors in the total HF energy.
Paper 5: Near–idempotent matrices  Two theorems are presented in this paper that can be used to locate the eigenvalues of a matrix that is near–idempotent. This paper will be extended with some additional ongoing work.
The objective of this thesis has been to enable quantum modeling of systems containing thousands of atoms with the Hartree–Fock and Kohn–Sham methods. This requires efficient methods for the Fock/Kohn–Sham and the density matrix constructions as these are the computationally demanding steps in Self–Consistent Field calculations for large systems. The focus in this thesis has been on an efficient representation of potentials and densities and on the density matrix construction. A number of methods can be used for the density matrix construction. These methods all rely on efficient linear algebra, in particular fast matrix–matrix multiplication. In this thesis, a hierarchic sparse matrix data structure was presented to reduce the memory usage in the entire SCF calculation and to facilitate the implementation of sparse algebraic operations. A truncation method was presented to enforce sparsity, for which the selected threshold can be related to desired accuracy in total energy or eigenvalues. Symmetry in matrices has also been utilized, for example to obtain fast symmetric matrix squares which is the computational kernel in trace correcting purification, one of the density matrix construction methods. The use of symmetry has reduced the computational time with more than 45% which is close to the theoretical 50%. A method to calculate the chemical potential at negligible cost when density matrix purification is used was also proposed. This method can be combined with the Spectral Transformation Lanczos method to obtain eigenvalues around the HOMO–LUMO gap.

Ongoing and future work  Paper 5 will be extended to include some additional ongoing work. The following are issues that will be attended in ongoing/future work;

• It will be possible to combine the method for the chemical potential with theorem 1 in paper 5 to calculate the HOMO and LUMO eigenvalues at negligible cost. This is useful since information about the HOMO and LUMO eigenvalues is valuable for SCF convergence accelerating methods[17]. It is also useful for density matrix construction
methods that require the chemical potential as input since this information can be used in the following SCF iteration.

- Systems with small HOMO–LUMO gaps are problematic in density matrix purification methods since many iterations are needed in order to separate the HOMO and LUMO eigenvalues. Also, since the accumulated error in purification is dominated by the inverse of the HOMO–LUMO gap[11], the result of the purification may even, for really small HOMO–LUMO gaps, be meaningless.

- The current implementation involves the use of “size–inconsistent” quantities like the Frobenius norm and the matrix trace. We plan to perform an analysis of the use of such quantities in calculations for large systems. Possibly, one has to avoid the use of these quantities.

- With truncation methods traditionally used in the SCF calculation, it is difficult to ensure that calculations in different parts of the SCF cycle are at the same level of accuracy. In this thesis, the matrix truncation has already been related to the norm of the matrix, which in turn can be related to desired accuracy in different properties. We plan to take this one step further by relating the truncation thresholds in all parts of the calculation to the requested accuracy in some appropriate matrix norm.
A.1 Definitions

The Fock matrix introduced in chapter 2 consists of one-electron $H_1$ and two-electron contributions $F_{2\text{el}}$.

$$F = H_1 + F_{2\text{el}}$$  \hspace{1cm} (A.1)

In the Hartree–Fock method $F_{2\text{el}} = J + K$, in the Kohn–Sham method $F_{2\text{el}} = J + F_{xc}$, and in hybrid methods $F_{2\text{el}} = J + \alpha K + F_{xc}$ for some scalar $\alpha$. These matrices can all be formulated without the $C$–matrix:

- The one electron matrix:

$$H_1 = -\int_{\mathbb{R}^3} x(r_1) \frac{\nabla^2 x^T(r_1)}{2} dr_1 - \int_{\mathbb{R}^3} x(r_1) \sum_A \frac{Z_A}{|r_A - r_1|} x^T(r_1) dr_1$$  \hspace{1cm} (A.2)

Here $Z_A$ and $r_A$ is the charge and position of atom $A$.

- The Coulomb matrix:

$$J = 2 \int \int_{\mathbb{R}^6} \frac{x(r_1)x^T(r_2)Px(r_2)x^T(r_1)}{|r_1 - r_2|} dr_2 dr_1$$  \hspace{1cm} (A.3)

$$K = \int \int_{\mathbb{R}^6} \frac{x(r_1)x^T(r_2)Px(r_1)x^T(r_2)}{|r_1 - r_2|} dr_2 dr_1$$  \hspace{1cm} (A.5)
• The KS exchange correlation matrix:

\[ F_{xc} = \int_{\mathbb{R}^3} x(r) \frac{\partial F}{\partial \rho} \bigg|_{\rho=\rho(r)} x^T(r) \, dr \]  

(A.7)

where \( \int F(\rho) \, dr \) is the energy functional which here for simplicity is assumed to be only density dependent.

The overlap matrix is defined as

\[ S = \int_{\mathbb{R}^3} x(r)x^T(r) \, dr \]  

(A.8)

### A.2 Implementation

A main part of this thesis concerns efficient matrix representations of electron density and potential operators. In this section more details about the actual implementation in C++ is given. I describe here; The use of proxy objects to avoid allocation of large temporary objects and copying of matrices. The use of templates to put the hierarchic, the Compressed Sparse Column (CSC), and the Blocked Compressed Sparse Column (BCSC) data structures into practice. Also, I compare the complexity of the matrix–matrix multiplication routines for the Compressed Sparse Column and the hierarchic data structures. It should be noted that the code snippets presented here does not form complete C++ classes, but serve only as illustrative examples.

#### A.2.1 Proxy objects

A proxy is usually used to provide a surrogate or placeholder for another object to control access to it. A proxy can for example be used to wait with, or completely avoid, the instantiation of objects until they are really required. The following proxy, for the multiplication of two objects of type \( T \), is used to avoid the instantiation of the temporary object usually returned from the multiplication operator.

```cpp
template<typename T>
struct XY {
    T const & A;
    T const & B;
    XY(T const & AA, T const & BB):A(AA), B(BB) {}  
};
```
The following multiplication operator for the multiplication of two objects of type T, returns a proxy. Note that the operator is declared inline, which means that there is no additional overhead associated with this design pattern.

```cpp
template<typename T>
inline XY<T> operator*(T const & A, T const & B) {
    return XY<T>(A, B);
}
```

The actual multiplication algorithm is described in the class of type T, for example in an assignment operator as described in section A.2.3. It is worth noting that synchronization problems do not appear since the life time of the proxy object is short.

### A.2.2 Matrix templates

A general matrix with elements of the type Telement can be described by a template. The elements may be of any data type that has the functionality needed by the member functions used in the instantiation.

```cpp
template<typename Telement>
class Matrix {
    Telement* elements;
};
```

A matrix in the CSC format can be defined in a similar way;

```cpp
template<typename Telement>
class Matrix_CSC {
    int* colpoint;
    int* rowind;
    Telement* elements;
};
```

These templates can be combined into new types to obtain the BCSC and hierarchic data structures. The hierarchic data structure has three levels in the following example.

```cpp
template<typename Treal>
typedef Matrix_CSC<Matrix<Treal> > Matrix_BCSC;
template<typename Treal>
typedef Matrix<Matrix<Matrix<Treal> > > Matrix_Hierarchic;
```
A.2.3 Matrix–matrix multiplication

The matrix–matrix multiplication routine for a general matrix is given below:

```cpp
template<typename Telement>
Matrix<Telement>& Matrix<Telement>::operator=(XY<Matrix<Telement>, Matrix<Telement> > const & MM) {
    Matrix<Telement>& C = *this;
    for (int colC = 0; colC < C.ncols; colC++)
        for (int rowC = 0; rowC < C.nrows; rowC++) {
            C(rowC, colC) = MM.A(rowC, 0) * MM.B(0, colC);
            for (int colA = 1; colA < A.ncols; colA++)
                C(rowC, colC) += MM.A(rowC, colA) * MM.B(colA, colC);
        }
    return C;
}
```

For the hierarchic data structure, specific routines for each level are not needed since this template can be used for all levels in the hierarchy. One may, however, wish to implement a specialization for the lowest level, since the performance critical operations are done there.

The matrix–matrix multiplication routine first suggested in ref. [36] for the CSC and BCSC data structures is described below. Note that the same lines of code is used for both data structures since the type of the matrix elements is left open until instantiation.

```cpp
template<typename Telement>
Matrix_CSC<Telement>& Matrix_CSC<Telement>::operator=(XY<Matrix_CSC<Telement>, Matrix_CSC<Telement> > const & MM) {
    Matrix_CSC<Telement>& C = *this;
    C.nrofel = 0;
    int* tmpint = new int[C.nrows];
    Telement* tmpcol = new Telement[C.nrows];
    for(int ind =0; ind < C.nrows; ind++)
        tmpint[ind] = -1;
    int rowA, rowB, rowC;
    for(int j = 0;j < C.ncols; j++) {
        C.colpoint[j] = C.nrofel;
        for(int indB = MM.B.colpoint[j]; indB < MM.B.colpoint[j+1]; indB++) {
            rowB = MM.B.rowind[indB];
```
for(int indA = MM.A.colpoint[rowB]; indA < MM.A.colpoint[rowB+1]; indA++) {
    rowA = A.rowind[indA];
    if(tmpint[rowA] != j) {
        tmpcol[rowA] = MM.A.elements[indA] * MM.B.elements[indB];
        C.rowind[C.nrofel] = rowA;
        C.nrofel++;
        tmpint[rowA] = j;
    } else {
        tmpcol[rowA] += MM.A.elements[indA] * MM.B.elements[indB];
    }
}
for (int indC = C.colpoint[j]; indC < C.nrofel; indC++) {
    rowC = C.rowind[indC];
    C.elements[indC] = tmpcol[rowC];
}
C.colpoint[C.nrofcols] = C.nrofel;
delete[] tmpint;
delete[] tmpcol;
return C;

The development and implementation of algorithms for the CSC/BCSC data structures is clearly more complicated than for the hierarchic data structure.

A.2.4 Supported operations in the Hierarchic Matrix Library

With the code presented above users of the hierarchic matrix library can perform matrix–matrix multiplication with the simple syntax $C = A \times B$. This kind of syntax gives more transparent source code than, for example, the commonly used $\text{dgemm}$ routine in BLAS with 13 arguments in the function call.

In the Hierarchic Matrix Library (HML) the following list of basic linear algebra routines (with the presented syntax) are supported. Three matrix types exist in HML; general, symmetric, and upper triangular. The matrices appearing in this list are of the general type, unless something else is stated and \text{op}(X) is either $X$ or $\text{transpose}(X)$. Note that this is not an exhaustive list of operations.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C = \alpha \ast \text{op}(A) \ast \text{op}(B)$</td>
<td></td>
</tr>
<tr>
<td>$C + = \alpha \ast \text{op}(A) \ast \text{op}(B)$</td>
<td></td>
</tr>
<tr>
<td>$C = \alpha \ast \text{op}(A) \ast \text{op}(B) + \beta \ast C$</td>
<td></td>
</tr>
<tr>
<td>$C = A + B$</td>
<td>A is symmetric</td>
</tr>
<tr>
<td>$B + = A$</td>
<td>A and $B$ are symmetric</td>
</tr>
<tr>
<td>$B + = \alpha \ast A$</td>
<td></td>
</tr>
<tr>
<td>$C = \alpha \ast A \ast B$</td>
<td></td>
</tr>
<tr>
<td>$C + = \alpha \ast A \ast B$</td>
<td>A is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast A \ast B + \beta \ast C$</td>
<td>A is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast B \ast A$</td>
<td>A is symmetric</td>
</tr>
<tr>
<td>$C + = \alpha \ast B \ast A$</td>
<td>A is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast B \ast A + \beta \ast C$</td>
<td>A is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast B \ast A + \beta \ast C$</td>
<td>A and $B$ are symmetric</td>
</tr>
<tr>
<td>$B = \alpha \ast \text{op}(A) \ast B$</td>
<td>A is upper triangular</td>
</tr>
<tr>
<td>$B = \alpha \ast \text{op}(A) \ast B$</td>
<td>A is upper triangular</td>
</tr>
<tr>
<td>$B + = \alpha \ast A$</td>
<td>A and $B$ are symmetric</td>
</tr>
<tr>
<td>$B + = A$</td>
<td>A and $B$ are symmetric</td>
</tr>
<tr>
<td>$C + = \alpha \ast A \ast \text{transpose}(A)$</td>
<td>C is symmetric</td>
</tr>
<tr>
<td>$C + = \alpha \ast A \ast A$</td>
<td>A and $C$ are symmetric</td>
</tr>
<tr>
<td>$C = A + B$</td>
<td>A, $B$, and $C$ are symmetric</td>
</tr>
<tr>
<td>$A = Z \ast A \ast \text{transpose}(Z)$</td>
<td>$Z$ is upper triangular and $A$ is symmetric</td>
</tr>
<tr>
<td>$A = \text{transpose}(Z) \ast A \ast Z$</td>
<td>$Z$ is upper triangular and $A$ is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast A \ast \text{transpose}(A)$</td>
<td>C is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast A \ast \text{transpose}(A) + \beta \ast C$</td>
<td>C is symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast A \ast A$</td>
<td>A and $C$ are symmetric</td>
</tr>
<tr>
<td>$C = \alpha \ast A \ast A + \beta \ast C$</td>
<td>A and $C$ are symmetric</td>
</tr>
</tbody>
</table>


