Quantum Chemistry for Large Systems

Elias Rudberg

Theoretical Chemistry
School of Biotechnology
Royal Institute of Technology
Stockholm 2007

Quantum Chemistry for Large Systems
Doctoral thesis

ISBN 978-91-7178-797-2
Printed by Universitetsservice US AB,
Stockholm, Sweden, 2007
Typeset in LaTeX by the author

Abstract

This thesis deals with quantum chemistry methods for large systems. In particular, the thesis focuses on the efficient construction of the Coulomb and exchange matrices which are important parts of the Fock matrix in Hartree–Fock calculations. Density matrix purification, which is a method used to construct the density matrix for a given Fock matrix, is also discussed.

The methods described are not only applicable in the Hartree–Fock case, but also in Kohn–Sham Density Functional Theory calculations, where the Coulomb and exchange matrices are parts of the Kohn–Sham matrix. Screening techniques for reducing the computational complexity of both Coulomb and exchange computations are discussed, including the fast multipole method, used for efficient computation of the Coulomb matrix.

The thesis also discusses how sparsity in the matrices occurring in Hartree–Fock and Kohn–Sham Density Functional Theory calculations can be used to achieve more efficient storage of matrices as well as more efficient operations on them.

Preface

The work presented in this thesis has been carried out at the Department of Theoretical Chemistry, School of Biotechnology, Royal Institute of Technology, Stockholm, Sweden.

List of papers included in the thesis

Paper 1 *Efficient implementation of the fast multipole method*, Elias Rudberg and Paweł Sałek, J. Chem. Phys. **125**, 084106 (2006).

Paper 2 *A hierarchic sparse matrix data structure for large-scale Hartree-Fock/Kohn-Sham calculations*, Emanuel H. Rubensson, Elias Rudberg, and Paweł Sałek, J. Comput. Chem. **28**, 2531 (2007).

Paper 3 *Nonlocal exchange interaction removes half-metallicity in graphene nanoribbons*, Elias Rudberg, Paweł Sałek, and Yi Luo, Nano Letters **7**, 2211 (2007).

Paper 4 *A linear scaling study of solvent-solute interaction energy of drug molecules in aqua solution*, Laban Bondesson, Elias Rudberg, Yi Luo, and Paweł Sałek, J. Phys. Chem. B **111**, 10320 (2007).

Paper 5 *Basis set dependence of solute-solvent interaction energy of benzene in water: a HF/DFT study*, Laban Bondesson, Elias Rudberg, Yi Luo, and Paweł Sałek, *Submitted manuscript*

Paper 6 *Rotations of occupied invariant subspaces in self–consistent field calculations*, Emanuel H. Rubensson, Elias Rudberg, and Paweł Sałek, *Submitted manuscript*

Paper 7 *Density matrix purification with rigorous error control*, Emanuel H. Rubensson, Elias Rudberg, and Paweł Sałek, *Submitted manuscript*

Paper 8 *Estimation of errors in Coulomb and exchange matrix construction*, Elias Rudberg, Emanuel H. Rubensson, and Paweł Sałek, *Manuscript in preparation*

Paper 9 *Hartree-Fock calculations with linearly scaling memory usage*, Elias Rudberg, Emanuel H. Rubensson, and Paweł Sałek, *Manuscript in preparation*

Paper 10 *Truncation of small matrix elements based on the Euclidean norm for blocked data structures*, Emanuel H. Rubensson, Elias Rudberg, and Paweł Sałek, *Manuscript in preparation*

List of papers not included in the thesis

- *Calculations of two-photon charge-transfer excitations using Coulomb-attenuated density-functional theory*, Elias Rudberg, Paweł Sałek, Trygve Helgaker, and Hans Ågren, J. Chem. Phys. **123**, 184108 (2005).

- *Heisenberg exchange in dinuclear manganese complexes: a density functional theory study*, Elias Rudberg, Paweł Sałek, Zilvinas Rinkevicius, and Hans Ågren, J. Chem. Theor. Comput. **2**, 981 (2006).

- *Sparse matrix algebra for quantum modeling of large systems*, Emanuel H. Rubensson, Elias Rudberg, and Paweł Sałek, Proceedings of PARA'06, Springer LNCS **4699**, 90 (2007).

Comments on my contribution to the papers included

- I was responsible for the development of algorithms, for the implementation, for the calculations and for the writing of Paper 1.
- I took part in development of ideas and was responsible for the calculations and for part of the writing of Paper 2.
- I was responsible for the calculations and for the writing of Paper 3.
- I was responsible for implementing necessary features in the program used for the calculations and assisted in the writing of Paper 4.
- I was responsible for implementing necessary features in the program used for the calculations and assisted in the writing of Paper 5.
- I assisted in method development and in the writing of Paper 6.
- I assisted in method development and in the writing of Paper 7.
- I was responsible for the calculations and for the writing of Paper 8.
- I was responsible for the calculations and for the writing of Paper 9.
- I assisted in method development and in the writing of Paper 10.

Acknowledgments

Many thanks to the following people:

Paweł Sałek for being an excellent supervisor. Hans Ågren for making it possible for me to work with this. Emanuel Rubensson, Zilvinas Rinkevicius, Yi Luo and Laban Bondesson for working with me. Peter Taylor for his support. Everyone at the Theoretical Chemistry group in Stockholm, for making it a good place to work at.

Special thanks to Peter Hammar who helped reading and correcting the thesis text.

This research has been supported by the Sixth Framework Programme Marie Curie Research Training Network under contract number MRTN-CT-2003-506842.

Contents

Part I

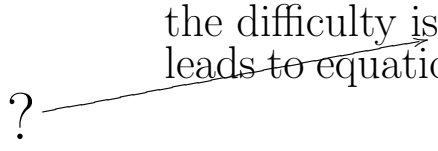
Introductory Chapters

Chapter 1

Introduction

Most chemists today agree that chemistry is in principle well described by the theory of Quantum Mechanics. Quantum Mechanics was developed during the 1920's. In 1929, P. A. M. Dirac wrote a paper including the following famous statement[1]:

> The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the application of these laws leads to equations much too complicated to be soluble.
>
> P. A. M. Dirac



This is a quite bold statement, which might seem provocative to a theoretical chemist if misinterpreted as "the whole of chemistry is known". That is not, of course, what is meant: the statement says something about the *underlying physical laws*. To solve real problems in theoretical chemistry, two things are necessary. Firstly, knowledge of the underlying physical laws. Secondly, ways to get meaningful results from the mathematical equations dictated by those laws. It is interesting that Dirac used the word "only" when referring to the task of solving the equations. This wording does not really add any information to the statement; it merely reflects the author's opinion that knowing the equations is more important than solving them.

Now, nearly 80 years later, the science of theoretical chemistry is still struggling with the solution of these equations. Therefore, one might argue that the word "only" should be omitted in a modern version of the above statement.

The mathematical equations in question can be "solved" in many different ways depending on which level of approximation is adopted. This thesis deals with one type of

"solutions": those given by the Hartree–Fock and Kohn–Sham Density Functional Theory methods. With these methods, large molecular systems can be treated quantum mechanically with the use of rather crude approximations.

Chapter 2

Hartree–Fock Theory

This chapter gives a brief description of the Hartree–Fock method from a computational point of view. A more thorough treatment of Hartree–Fock theory, including a complete derivation, can be found in the book by Szabo and Ostlund [2].

2.1 The Schrödinger Equation

The algorithms discussed in this thesis are useful when quantum mechanics is applied to the physical problem of N identical charged particles (the electrons) in the field of M fixed classical point charges (the nuclei). In the most common formulation of quantum mechanics, solving this problem means solving the *N-electron Schrödinger equation*:

$$\hat{H}\psi = E\psi \tag{2.1}$$

The Schrödinger equation (2.1) is an eigenvalue equation involving the Hamiltonian operator $\hat{H}$, the wave function ψ, and the eigenvalue E. Here, we will focus on finding the ground state of the system; that is, the eigenfunction ψ that gives the lowest energy E. The Hamiltonian operator $\hat{H}$ is given by

$$\hat{H} = -\sum_{i=1}^{N} \frac{1}{2}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M} \frac{Z_A}{r_{iA}} + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \frac{1}{r_{ij}} \tag{2.2}$$

The wave function ψ is a function of the spatial positions and spins of the electrons,

$$\psi = \psi(\mathbf{r}_1, \omega_1, \mathbf{r}_2, \omega_2, \ldots, \mathbf{r}_N, \omega_N) \tag{2.3}$$

with the vectors $\mathbf{r}_i = (x_i, y_i, z_i)$ being the position of electron i and ω_i being the spin of electron i : ψ is a function of $4N$ variables. In the definition of the Hamiltonian operator

(2.2), ∇_i^2 is the Laplacian operator

$$\nabla_i^2 = \frac{\partial^2}{\partial x_i^2} + \frac{\partial^2}{\partial y_i^2} + \frac{\partial^2}{\partial z_i^2} \tag{2.4}$$

Z_A is the charge of nucleus A, $r_{iA} = |\mathbf{r}_i - \mathbf{r}_A|$ is the distance between electron i and nucleus A, and $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the distance between electron i and electron j. The spin variables ω_i do not appear in the Hamiltonian operator, but become important because of the Pauli exclusion principle, which states that *the wave function ψ must be antisymmetric with respect to the interchange of any two electrons*:

$$\psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_N) = -\psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N) \tag{2.5}$$

Here, the notation $\mathbf{x}_i = (\mathbf{r}_i, \omega_i)$ is used for the combination of spatial and spin variables of electron i.

To summarize, the wave function ψ must satisfy two requirements: firstly, it must fulfill the Schrödinger equation (2.1), and secondly it must fulfill the antisymmetry condition (2.5).

2.2 Slater Determinants

As described in section 2.1, the wave function must satisfy two distinct conditions. It is then natural to try to satisfy one condition first, and then see what can be done about the other condition. In the approach described in the following, one first makes sure the antisymmetry condition is fulfilled.

Now, how can one construct a function ψ that satisfies the antisymmetry condition (2.5)? For a case with just two electrons, a possible choice of ψ is

$$\psi(\mathbf{x}_1, \mathbf{x}_2) = \phi_1(\mathbf{x}_1)\phi_2(\mathbf{x}_2) - \phi_1(\mathbf{x}_2)\phi_2(\mathbf{x}_1) \tag{2.6}$$

where $\phi_i(\mathbf{x})$ are one–electron wave functions, i.e. functions of the space and spin variables of just one electron. More generally, an antisymmetric function ψ can be achieved by taking linear combinations of products of one–electron functions using the combinatorics of a matrix determinant:

$$\psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \begin{vmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_N(\mathbf{x}_N) \end{vmatrix} \tag{2.7}$$

A function ψ constructed according to (2.7) is called a *Slater determinant*. A Slater determinant will always satisfy the antisymmetry condition (2.5), for any choice of the one–electron functions $\phi_i(\mathbf{x})$. Next, we note that any linear combination of Slater determinants will also satisfy (2.5). For this reason, Slater determinants are very useful as basic building blocks when one tries to construct a solution to the Schrödinger equation (2.1) that at the same time satisfies the antisymmetry condition (2.5). This thesis focuses on the computationally simplest option: to use a single Slater determinant, varying the one–electron functions $\phi_i(\mathbf{x})$ to get as near as possible to a solution of (2.1). This is known as the *Hartree–Fock* approximation.

2.3 Basis Sets

When working with Slater determinants, everything is based on the one–electron functions $\phi_i(\mathbf{x})$, usually taken to be products of a spatial part and a spin part:

$$\phi_i(\mathbf{x}) = \varphi_i(\mathbf{r})\gamma_i(\omega) \tag{2.8}$$

The spin part $\gamma(\omega)$ is completely described using just two functions $\alpha(\omega)$ and $\beta(\omega)$. The spatial part $\varphi_i(\mathbf{r})$, on the other hand, could in principle be *any* function of the three spatial coordinates. One therefore constructs $\varphi_i(\mathbf{r})$ as a linear combination of *basis functions*:

$$\varphi_i(\mathbf{r}) = \sum_{j=1}^{n} c_{ij} b_j(\mathbf{r}) \tag{2.9}$$

The set of basis functions $\{b_j(\mathbf{r})\}$ is often called a *basis set*. In principle, allowing $\varphi_i(\mathbf{r})$ complete flexibility to take any functional form would require an infinite number of basis functions. In practice, however, one has to settle for a finite number of basis functions. This thesis focuses on one particular type of basis functions: the Gaussian Type Linear Combination of Atomic Orbital (GT–LCAO) type of functions. The primitive functional form used is:

$$h_k(\mathbf{r}) = e^{-\alpha_k(\mathbf{r}-\mathbf{r_0})^2} \tag{2.10}$$

where α is referred to as the exponent of the primitive function, and $\mathbf{r_0}$ is the spatial point around which the function is centered. Given primitive functions of the type (2.10), the basis functions $b_j(\mathbf{r})$ are constructed as linear combinations of the primitive functions, multiplied by some polynomials in the displacement coordinates $\mathbf{r} - \mathbf{r_0}$:

$$b(\mathbf{r}) = P(\mathbf{r} - \mathbf{r_0}) \sum_k g_k h_k(\mathbf{r}) \tag{2.11}$$

The center points $\mathbf{r_0}$ of the basis functions are usually set to the nuclear positions and the polynomials P are chosen as the real solid harmonics[7]. The exponents α and expansion coefficients g_k are chosen to allow as good a description as possible of the one–electron functions $\phi_i(\mathbf{x})$ using only a limited number of basis functions.

An important property of GT–LCAO basis functions is that they are *localized*. That is, each basis function is nonzero only in a limited region of space. This is because the primitive Gaussian functions in (2.10) decay exponentially with the distance from the center point $\mathbf{r_0}$. Strictly speaking, the function value is not exactly zero even for large distances, but the exponential decay means that as the distance from $\mathbf{r_0}$ increases the function will soon become negligible. In practice, it works fine to assume that each basis function is zero outside some radius from the center point, as long as that radius is chosen carefully.

2.4 The Hartree–Fock Method

As was mentioned in Section 2.2, the use of a single Slater determinant as an approximate wave function is known as the *Hartree–Fock* approximation. There are a few different variants of this approximation depending on how one treats the spins of the electrons that make up the Slater determinant. The most widely used variant is called *Restricted Hartree–Fock* (RHF). In RHF, one makes use of the fact that electrons tend to stay in pairs; a "spin–up" electron together with a "spin–down" electron both having the same spatial part of the one–electron wave function. A molecular system with an even number of electrons that are all paired in this way is called a *closed shell* system. Many systems studied with quantum chemistry methods are quite well described by a closed shell Slater determinant, and therefore the RHF method is commonly used. The computational details of the RHF method are given in Section 2.4.1. In Section 2.4.3, an alternative known as *Unrestricted Hartree–Fock* (UHF) is described, which is useful in cases when all electrons are not sitting in pairs.

2.4.1 Restricted Hartree–Fock

This section describes the RHF computational procedure for a molecular system with N electrons and n basis functions. The following three $n \times n$ matrices are important: the overlap matrix $\mathbf{S}$, the density matrix $\mathbf{D}$, and the Fock matrix $\mathbf{F}$. The overlap matrix is given by

$$S_{pq} = \int b_p(\mathbf{r}) b_q(\mathbf{r}) d\mathbf{r} \tag{2.12}$$

Before discussing the density matrix, consider the single Slater determinant used to approximate the wave function. In RHF, all electrons are paired; this means there are only $\frac{N}{2}$ different spatial functions for the N electrons. These spatial functions are known as *orbitals*. Each orbital φ_i is defined as a linear combination of basis functions, as stated in equation (2.9). The density matrix $\mathbf{D}$ is related to the coefficients c_{ij} of equation (2.9):

$$D_{pq} = 2\sum_{a=1}^{\frac{N}{2}} c_{ap}c_{aq} \tag{2.13}$$

Using the density matrix $\mathbf{D}$, the total electron density $\rho(\mathbf{r})$ can be written as

$$\rho(\mathbf{r}) = \sum_{pq} D_{pq}b_p(\mathbf{r})b_q(\mathbf{r}) \tag{2.14}$$

hence the name "density matrix". The Fock matrix $\mathbf{F}$ is given by

$$\mathbf{F} = \mathbf{H}_{\text{core}} + \mathbf{G} \tag{2.15}$$

where

$$\mathbf{H}_{\text{core}} = \mathbf{T} + \mathbf{V} \tag{2.16}$$

with the kinetic energy term $\mathbf{T}$ and the electron-nuclear attraction term $\mathbf{V}$ given by

$$T_{pq} = -\frac{1}{2}\int b_p(\mathbf{r})\nabla^2 b_q(\mathbf{r})d\mathbf{r} \tag{2.17}$$

$$V_{pq} = -\int b_p(\mathbf{r})\left[\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}-\mathbf{r}_A|}\right]b_q(\mathbf{r})d\mathbf{r} \tag{2.18}$$

The matrix $\mathbf{G}$ is the two-electron part of the Fock matrix, defined using the density matrix as

$$G_{pq} = \sum_{rs} D_{rs}\big[(pq|rs) - \frac{1}{2}(pr|sq)\big] \tag{2.19}$$

The numbers $(pq|rs)$ and $(pr|sq)$ are known as two-electron integrals:

$$(pq|rs) = \int \frac{b_p(\mathbf{r_1})b_q(\mathbf{r_1})b_r(\mathbf{r_2})b_s(\mathbf{r_2})}{|\mathbf{r_1}-\mathbf{r_2}|}d\mathbf{r_1}d\mathbf{r_2} \tag{2.20}$$

The three different contributions $\mathbf{T}$, $\mathbf{V}$, and $\mathbf{G}$ to the Fock matrix $\mathbf{F}$ each corresponds to one of the terms in the Hamiltonian operator of equation (2.2). The matrices $\mathbf{T}$ and $\mathbf{V}$ depend only on basis set and nuclear positions; therefore $\mathbf{H}_{\text{core}}$ is independent of $\mathbf{D}$. Given a set of orbitals and a corresponding density matrix $\mathbf{D}$, a Fock matrix $\mathbf{F}$ can be computed. Then the energy corresponding to that particular set of orbitals can be computed as

$$E_{\text{RHF}} = \text{Tr}\big(\mathbf{DH}_{\text{core}}\big) + \frac{1}{2}\text{Tr}\big(\mathbf{DG}\big) \tag{2.21}$$

where the notation $\mathrm{Tr}(\mathbf{A})$ stands for the *trace* of the matrix: $\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{n} A_{ii}$. Using (2.15) the energy can equivalently be written using the Fock matrix:

$$E_{\mathrm{RHF}} = \frac{1}{2}\mathrm{Tr}\big(\mathbf{D}\mathbf{H}_{\mathrm{core}}\big) + \frac{1}{2}\mathrm{Tr}\big(\mathbf{D}\mathbf{F}\big) \tag{2.22}$$

In a way, this is all we need to know in order to find the RHF electron density and corresponding energy; if we could try all possible choices of $\mathbf{D}$, the one with lowest energy is the RHF solution. In practice, however, it is not possible to try all possible choices of $\mathbf{D}$. Some kind of optimization scheme is needed. Section 2.4.2 describes one such scheme, known as the *Self–Consistent Field procedure* (SCF).

2.4.2 The Self–Consistent Field Procedure

The Self–Consistent Field procedure (SCF) is an iterative method for finding the density matrix that gives the lowest energy. It consists of two main steps, here labeled ($\mathbf{D} \to \mathbf{F}$) and ($\mathbf{F} \to \mathbf{D}$). In the ($\mathbf{D} \to \mathbf{F}$) step, a new Fock matrix $\mathbf{F}$ is constructed from $\mathbf{D}$ using equation (2.15). The elements of $\mathbf{D}$ enter through equation (2.19). In the ($\mathbf{F} \to \mathbf{D}$) step, a new density matrix is constructed for a given Fock matrix, in the following way: first, the generalized eigenvalue problem

$$\mathbf{F}\mathbf{C} = \mathbf{S}\mathbf{C}\mathbf{\Lambda} \tag{2.23}$$

is solved, giving the matrix of eigenvectors $\mathbf{C}$ and corresponding eigenvalues in the diagonal matrix $\mathbf{\Lambda}$. Then the new density matrix is created as

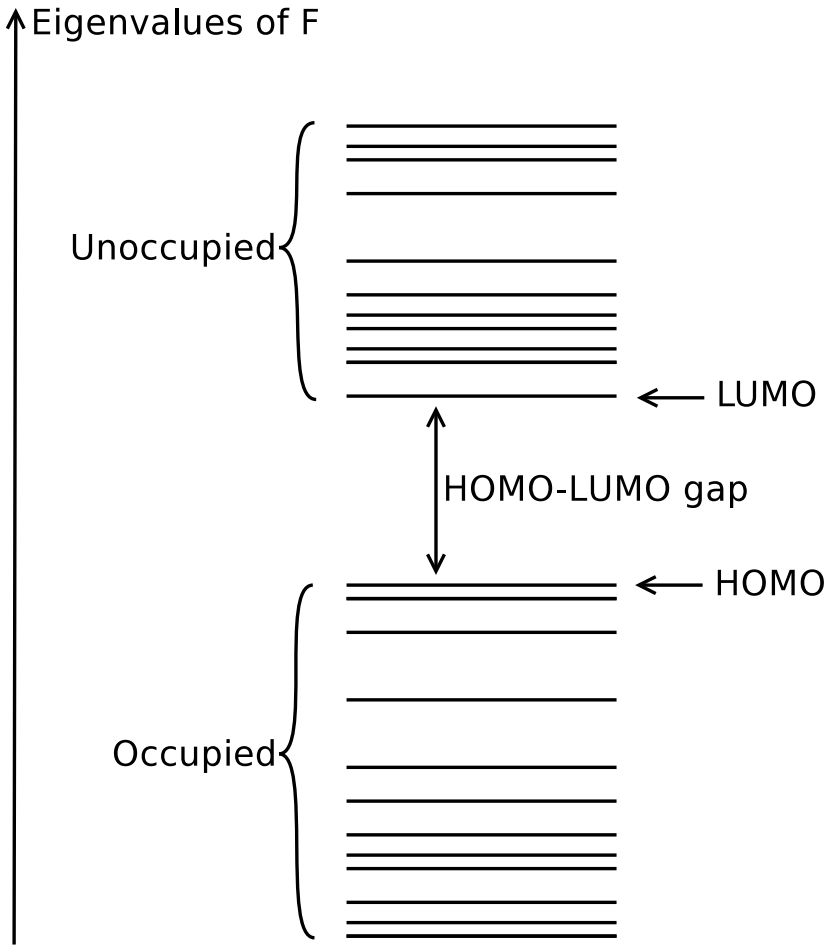
$$D_{pq} = 2\sum_{a=1}^{\frac{N}{2}} C_{pa}C_{qa} \tag{2.24}$$

Note that equation (2.24) is essentially the same as equation (2.13): the eigenvectors of $\mathbf{F}$ are interpreted as orbital coefficients.

When forming a new density matrix according to (2.24), one uses the $\frac{N}{2}$ eigenvectors of $\mathbf{F}$ that correspond to the smallest eigenvalues; the corresponding orbitals are referred to as *occupied orbitals*, while the orbitals associated with larger eigenvalues are called *unoccupied orbitals*. For the SCF procedure to be well–defined, there must be a gap between the eigenvalues of the *highest occupied molecular orbital* (HOMO) and the *lowest unoccupied molecular orbital* (LUMO), see Figure 2.1. This gap is referred to as the *HOMO–LUMO gap*. If the HOMO–LUMO gap is too small, the SCF procedure becomes difficult to converge.

Now that we know how to carry out the two main steps ($\mathbf{D} \to \mathbf{F}$) and ($\mathbf{F} \to \mathbf{D}$), the SCF procedure is really simple: first get a starting guess for $\mathbf{D}$. This guess could in principle be

Figure 2.1: Schematic illustration of occupied and unoccupied eigenvalues of the Fock matrix **F**. If the HOMO–LUMO gap is too small, the self–consistent field procedure is difficult to converge since it is then not clear which eigenvectors should be occupied in equation (2.24).



anything, but it helps if it is close to the final solution. Then compute a new **F** from **D**, a new **D** from that **F**, and so on, until **D** does not change any more. At that point one says that a *self–consistent field* has been found, hence the name SCF.

Although the two main steps of the SCF calculation ($\mathbf{D} \rightarrow \mathbf{F}$) and ($\mathbf{F} \rightarrow \mathbf{D}$) are straightforward to define, there are many different ways to carry them out. In later chapters we will discuss methods to efficiently do the ($\mathbf{D} \rightarrow \mathbf{F}$) and ($\mathbf{F} \rightarrow \mathbf{D}$) calculations, especially for large systems when localized basis functions are used.

2.4.3 Unrestricted Hartree–Fock

In the Unrestricted Hartree–Fock (UHF) method, one does not force the electrons to stay in pairs. Instead, each electron has its own spatial orbital φ_i; this means that there are now N different spatial orbitals φ_i instead of $\frac{N}{2}$ as in the RHF case. We will use the symbols N_α and N_β for the number of electrons of α and β spin, with $N_\alpha + N_\beta = N$. There are now separate density matrices $\mathbf{D}^\alpha$ and $\mathbf{D}^\beta$ for the α and β electrons, respectively:

$$D^\alpha_{pq} = \sum_{a=1}^{N_\alpha} c^\alpha_{ap} c^\alpha_{aq} \tag{2.25}$$

$$D^\beta_{pq} = \sum_{a=1}^{N_\beta} c^\beta_{ap} c^\beta_{aq} \tag{2.26}$$

It is also useful to define the total density matrix $\mathbf{D}^{tot}$ as

$$\mathbf{D}^{tot} = \mathbf{D}^\alpha + \mathbf{D}^\beta \tag{2.27}$$

Now, UHF Fock matrices $\mathbf{F}^\alpha$ and $\mathbf{F}^\beta$ are defined as

$$\mathbf{F}^\alpha = \mathbf{H}_{\text{core}} + \mathbf{G}^\alpha \tag{2.28}$$

$$\mathbf{F}^\beta = \mathbf{H}_{\text{core}} + \mathbf{G}^\beta \tag{2.29}$$

where the two-electron parts $\mathbf{G}^\alpha$ and $\mathbf{G}^\beta$ are given by

$$G^\alpha_{pq} = \sum_{rs} \left[D^{tot}_{rs} (pq|rs) - D^\alpha_{rs} (pr|sq) \right] \tag{2.30}$$

$$G^\beta_{pq} = \sum_{rs} \left[D^{tot}_{rs} (pq|rs) - D^\beta_{rs} (pr|sq) \right] \tag{2.31}$$

The matrix $\mathbf{H}_{\text{core}}$ is the same as in the RHF case. Given $\mathbf{D}^\alpha$ and $\mathbf{D}^\beta$ and the resulting Fock matrices $\mathbf{F}^\alpha$ and $\mathbf{F}^\beta$, the energy is given by

$$E_{\text{UHF}} = \text{Tr}\big(\mathbf{D}^{tot}\mathbf{H}_{\text{core}}\big) + \frac{1}{2}\text{Tr}\big(\mathbf{D}^\alpha\mathbf{G}^\alpha\big) + \frac{1}{2}\text{Tr}\big(\mathbf{D}^\beta\mathbf{G}^\beta\big) \tag{2.32}$$

The RHF method can be seen as a special case of UHF: if the number of electrons is even, with $N_\alpha = N_\beta = \frac{N}{2}$, and each α orbital has an identical β orbital corresponding to it, then $\mathbf{D}^\alpha = \mathbf{D}^\beta = \frac{1}{2}\mathbf{D}$, and consequently $\mathbf{F}^\alpha = \mathbf{F}^\beta = \mathbf{F}$. In this case the UHF energy expression (2.32) reduces to the RHF variant (2.21).

Instead of working with the density matrices $\mathbf{D}^\alpha$ and $\mathbf{D}^\beta$ directly, one can choose to use the total density matrix $\mathbf{D}^{tot}$ and the spin density matrix $\mathbf{D}^\sigma$.

$$\mathbf{D}^{tot} = \mathbf{D}^\alpha + \mathbf{D}^\beta \tag{2.33}$$

$$\mathbf{D}^\sigma = \mathbf{D}^\alpha - \mathbf{D}^\beta \tag{2.34}$$

Together, $\mathbf{D}^{tot}$ and $\mathbf{D}^\sigma$ contain the same information as $\mathbf{D}^\alpha$, $\mathbf{D}^\beta$. In the special case of RHF, $\mathbf{D}^\sigma = 0$. The spin density $\rho_\sigma(\mathbf{r})$ can be computed from $\mathbf{D}^\sigma$ as

$$\rho_\sigma(\mathbf{r}) = \sum_{pq} D^\sigma_{pq} b_p(\mathbf{r}) b_q(\mathbf{r}) \tag{2.35}$$

The spin density $\rho_\sigma(\mathbf{r})$ is positive in places where the α density dominates over the β density, and negative in places where the β density dominates. Plots of the spin density are useful as a way to visualize results of unrestricted calculations, as in Paper 3 of this thesis. The spin density plots shown there were generated by explicitly computing the spin density in each point according to (2.35).

The self–consistent field procedure for UHF is analogous to the RHF case; the two main steps are now $(\mathbf{D}^\alpha, \mathbf{D}^\beta \rightarrow \mathbf{F}^\alpha, \mathbf{F}^\beta)$ and $(\mathbf{F}^\alpha, \mathbf{F}^\beta \rightarrow \mathbf{D}^\alpha, \mathbf{D}^\beta)$. The $(\mathbf{F}^\alpha, \mathbf{F}^\beta \rightarrow \mathbf{D}^\alpha, \mathbf{D}^\beta)$ step is carried out by solving generalized eigenvalue problems analogous to (2.23) for α and β separately:

$$\mathbf{F}^\alpha \mathbf{C}^\alpha = \mathbf{S}\mathbf{C}^\alpha \mathbf{\Lambda}^\alpha \tag{2.36}$$

$$\mathbf{F}^\beta \mathbf{C}^\beta = \mathbf{S}\mathbf{C}^\beta \mathbf{\Lambda}^\beta \tag{2.37}$$

and then using the resulting eigenvectors $\mathbf{C}^\alpha$ and $\mathbf{C}^\beta$ as orbital coefficients when forming new density matrices according to (2.25) and (2.26).

Chapter 3

Density Functional Theory

This chapter deals with a computationally cheap way of improving upon the Hartree–Fock approximation: the Kohn–Sham formulation of Density Functional Theory (KS–DFT) [3]. Density Functional Theory is based on the Hohenberg–Kohn theorems [4], which essentially state that the electronic density alone contains enough information to serve as the basic quantity describing a N-electron molecular system. When the Kohn–Sham formulation of DFT is used, the electronic density is assumed to come from a wave function having the form of a single Slater determinant, just as in Hartree–Fock theory. This approach allows for a good description of the kinetic energy of the electrons, something that is difficult to achieve using other formulations of DFT.

The KS–DFT computational scheme is the same as the HF scheme, except for two things: there is a DFT exchange–correlation contribution to the energy (E_{XC}), and the Fock matrix $\mathbf{F}$ is replaced by the Kohn–Sham matrix $\mathbf{F^{KS}}$. Just as for HF, there are restricted and unrestricted versions of KS–DFT; in the unrestricted case there are two Kohn–Sham matrices $\mathbf{F}_\alpha^{\mathbf{KS}}$ and $\mathbf{F}_\beta^{\mathbf{KS}}$. In the following we will focus on the restricted case.

Before defining $\mathbf{F^{KS}}$, we need to introduce the Coulomb matrix $\mathbf{J}$ and the exchange matrix $\mathbf{K}$ as the two parts of the Hartree–Fock two-electron matrix $\mathbf{G}$:

$$\mathbf{G} = \mathbf{J} + \mathbf{K} \tag{3.1}$$

where

$$J_{pq} = \sum_{rs} D_{rs}(pq|rs) \tag{3.2}$$

$$K_{pq} = -\frac{1}{2}\sum_{rs} D_{rs}(pr|sq) \tag{3.3}$$

With the definitions (3.2) and (3.3), (3.1) is equivalent with the previous definition of $\mathbf{G}$

(2.19). Now, the DFT Kohn–Sham matrix $\mathbf{F^{KS}}$ is defined as

$$\mathbf{F^{KS}} = \mathbf{H}_{\text{core}} + \mathbf{J} + \theta\mathbf{K} + \mathbf{F^{XC}} \tag{3.4}$$

where θ is an empirically determined parameter and $\mathbf{F^{XC}}$ is the DFT *exchange–correlation* matrix. The KS–DFT energy is given by

$$E = \text{Tr}\big(\mathbf{DH}_{\text{core}}\big) + \frac{1}{2}\text{Tr}\Big(\mathbf{D}\big[\mathbf{J} + \theta\mathbf{K}\big]\Big) + E_{XC} \tag{3.5}$$

Note that if we set $E_{XC} = 0$ and $\theta = 1$, the DFT energy expression (3.5) reduces to the HF energy expression (2.21). The matrix $\mathbf{F^{XC}}$ and the exchange–correlation contribution to the energy E_{XC} are determined by the chosen *exchange–correlation functional* together with the density matrix $\mathbf{D}$. The chosen functional also determines the parameter θ of equation (3.4). Here, we will assume that the Generalized Gradient Approximation (GGA) is used. In this approximation, the functional is specified as an analytic function of the density ρ and the gradient of the density $\nabla\rho$. We will write this function as $F(\rho, \nabla\rho)$. Given the functional, as described by $F(\rho, \nabla\rho)$, E_{XC} and $\mathbf{F^{XC}}$ are given by

$$E_{XC} = \int F\big(\rho(\mathbf{r}), \nabla\rho(\mathbf{r})\big)\, d\mathbf{r} \tag{3.6}$$

$$F_{pq}^{XC} = \int \frac{\partial F}{\partial \rho}\bigg|_{\rho(\mathbf{r}),\nabla\rho(\mathbf{r})} b_p(\mathbf{r}) b_q(\mathbf{r})\, d\mathbf{r} + \int \frac{\partial F}{\partial(\nabla\rho)}\bigg|_{\rho(\mathbf{r}),\nabla\rho(\mathbf{r})} \nabla\big(b_p(\mathbf{r}) b_q(\mathbf{r})\big)\, d\mathbf{r} \tag{3.7}$$

The function $F(\rho, \nabla\rho)$ is given as an analytical expression, from which the functions $\frac{\partial F}{\partial \rho}$ and $\frac{\partial F}{\partial(\nabla\rho)}$ can also be derived analytically. The integrals in (3.6) and (3.7) are then evaluated numerically. In the integration, the elements of $\mathbf{D}$ enter through the evaluation of $\rho(\mathbf{r})$ and $\nabla\rho(\mathbf{r})$. Because the function $F(\rho, \nabla\rho)$ is usually quite complicated, no complete examples of such analytic expressions will be given here. In any case, it is an analytic expression containing terms like $\rho^{\frac{4}{3}}$ and $\log(\rho^{-\frac{1}{3}})$.

The different DFT exchange–correlation functionals can be divided into two main groups: *pure* DFT functionals and *hybrid* DFT functionals. The pure functionals have no $\mathbf{K}$ contribution to $\mathbf{F^{KS}}$; this corresponds to the parameter θ being zero. The hybrid functionals have nonzero θ. The most widely used hybrid functional, Becke's three–parameter functional with Lee-Young-Parr correlation part (B3LYP) [5], uses $\theta = 0.2$. The three B3LYP parameters, including θ, have been determined empirically; other values of these parameters are preferable in some cases[6].

From a computational point of view, a KS–DFT calculation is similar to a HF calculation. The SCF procedure can be used in the same way as in a HF calculation, with the two main

steps being ($\mathbf{D} \rightarrow \mathbf{F}^{\mathbf{KS}}$) and ($\mathbf{F}^{\mathbf{KS}} \rightarrow \mathbf{D}$). In the ($\mathbf{D} \rightarrow \mathbf{F}^{\mathbf{KS}}$) step, the Coulomb matrix $\mathbf{J}$ and the exchange matrix $\mathbf{K}$ must be computed just as in a HF calculation, and combined with the exchange–correlation matrix $\mathbf{F}^{\mathbf{XC}}$ according to (3.4). The ($\mathbf{F}^{\mathbf{KS}} \rightarrow \mathbf{D}$) step in a KS–DFT SCF calculation is precisely the same as the HF ($\mathbf{F} \rightarrow \mathbf{D}$) step. Because of the many similarities in HF and KS–DFT, it makes sense to implement both in the same computer program.

Integral Evaluation

In a HF/DFT calculation, much computer time is spent on the task of computing a new Fock/Kohn–Sham matrix for a given density matrix. Much of the work comes down to evaluating two–electron integrals of the type

$$(pq|rs) = \int \frac{b_p(\mathbf{r_1})b_q(\mathbf{r_1})b_r(\mathbf{r_2})b_s(\mathbf{r_2})}{|\mathbf{r_1}-\mathbf{r_2}|} d\mathbf{r_1} d\mathbf{r_2} \tag{4.1}$$

There are very many such integrals; if none of them are neglected, the number of integrals that must be evaluated is n^4, where n is the number of basis functions. Fortunately, there are ways to reduce the needed computational effort. There is some symmetry to take advantage of: it follows from (4.1) that $(pq|rs) = (qp|rs) = (pq|sr) = (rs|pq)$ etc., so that if all four indexes are different, there are eight integrals having the same value. This means that the number of unique integrals is about $\frac{1}{8}n^4$. For large systems, it is possible to reduce the number of integrals that need to be computed even more, provided that localized basis functions are used.

4.1 Primitive Gaussian Integrals

When the GT–LCAO type of basis functions are used, each two–electron integral $(pq|rs)$ is a sum of primitive integrals of the type

$$\int \frac{\Psi_A(\mathbf{r_1})\Psi_B(\mathbf{r_2})}{|\mathbf{r_1}-\mathbf{r_2}|} d\mathbf{r_1} d\mathbf{r_2} \tag{4.2}$$

where

$$\Psi_A(\mathbf{r}) = (x-x_A)^{i_A}(y-y_A)^{j_A}(z-z_A)^{k_A} e^{-\alpha_A(\mathbf{r}-\mathbf{r_A})^2} \tag{4.3}$$

$$\Psi_B(\mathbf{r}) = (x-x_B)^{i_B}(y-y_B)^{j_B}(z-z_B)^{k_B} e^{-\alpha_B(\mathbf{r}-\mathbf{r_B})^2} \tag{4.4}$$

At this level, each primitive distribution Ψ_A is determined by its exponent α_A, its center coordinates $\mathbf{r_A} = (x_A, y_A, z_A)$, and the three integers i_A, j_A and k_A.

Primitive integrals of the type (4.2) can be efficiently computed using the recurrence relations of the McMurchie–Davidson scheme. A thorough explanation of this scheme is found in the book by Helgaker et al [7].

4.2 Computational Scaling

When discussing integral evaluation in HF and DFT, one often talks about the computational *scaling* or *complexity* of different algorithms. The idea is that if one considers systems of different size but otherwise similar properties, using the same type of basis set (so that the number of basis functions grows linearly with the number of atoms), the performance of a particular algorithm will usually be rather predictable; the computational time T is often assumed to be of the form

$$T(n) = \text{const} \times n^p \tag{4.5}$$

where p is some number defining the scaling behavior. One then says that the algorithm has $\mathcal{O}(n^p)$ scaling. For example, a naive Fock matrix construction routine, where all integrals $(pq|rs)$ are computed explicitly, will scale as $\mathcal{O}(n^4)$. Here, the number of basis functions n was chosen as the variable used to express the scaling behavior; one could equivalently use any other quantity that increases linearly with the system size, such as the number of atoms or the number of electrons.

4.3 Screening

In this section, the concept of *screening* is introduced in the context of integral evaluation. What is meant by screening in this context is something like "neglect of small contributions". As was seen in Chapter 2, the elements of the two–electron matrix $\mathbf{G}$ are sums in which each term is of the form

$$D_{ab}(pq|rs) \tag{4.6}$$

Now, if it is known that $|D_{ab}| < \epsilon_1$ and that $|(pq|rs)| < \epsilon_2$, then the contribution to $\mathbf{G}$ is surely smaller than $\epsilon_1\epsilon_2$. The idea of integral screening is to systematically estimate bounds on $|D_{ab}|$ and $|(pq|rs)|$ and use that information to skip evaluation of $|(pq|rs)|$ whenever it is known that the resulting contribution to $\mathbf{G}$ would be smaller than some threshold value τ.

4.4 Cauchy–Schwarz Screening

The Schwarz inequality, also known as the Cauchy–Schwarz inequality, is very useful for providing bounds on two–electron integrals[8]:

$$|(pq|rs)| \leq \sqrt{(pq|pq)}\sqrt{(rs|rs)} \tag{4.7}$$

The quantity $\sqrt{(pq|pq)}$ associated with a pair of basis functions $b_p(\mathbf{r})$ and $b_q(\mathbf{r})$ is therefore of interest. Let D_{max} be the largest absolute density matrix element, and let C_{max} be the largest Cauchy-Schwarz factor $\sqrt{(pq|pq)}$ among all pairs of basis functions. Then a limit for the largest possible contribution to $\mathbf{G}$ from a particular basis function pair pq is given by $D_{max}C_{max}\sqrt{(pq|pq)}$. Therefore, once D_{max} and C_{max} are known, many basis function pairs can be neglected; it makes sense to create a list of all non–negligible basis function pairs. When the GT–LCAO type of basis functions are used, the number of non–negligible basis function pairs will scale as $\mathcal{O}(n)$, thanks to locality of basis functions. Using such a list of non–negligible basis function pairs as a starting point, all non–negligible integrals $(pq|rs)$ can easily be computed with $\mathcal{O}(n^2)$ complexity.

4.5 The Coulomb and Exchange Matrices

In the previous section, it was discussed how the formal $\mathcal{O}(n^4)$ scaling can be improved to $\mathcal{O}(n^2)$ while still considering all contributions to $\mathbf{G}$ on an equal footing. Further improvements can be made if one takes into account the way that density matrix elements are combined with the two–electron integrals. This differs between the Coulomb contribution $\mathbf{J}$ and the exchange contribution $\mathbf{K}$, see (3.2) and (3.3):

$$D_{rs}(pq|rs) \quad \text{(Coulomb)} \tag{4.8}$$

$$D_{rs}(pr|sq) \quad \text{(exchange)} \tag{4.9}$$

Therefore, when attempting to improve the scaling beyond $\mathcal{O}(n^2)$, it is helpful to use separate algorithms for the two contributions $\mathbf{J}$ and $\mathbf{K}$. In the case of $\mathbf{J}$, one can exploit the fact that pairs of basis functions together with the corresponding density matrix elements can be seen as a charge distribution for which a simplified description can be generated. In the case of $\mathbf{K}$, improved scaling can be achieved if density matrix elements decay with the distance between the corresponding basis functions, as is normally the case for non–metallic systems. Some more details about Coulomb matrix construction will be given in Chapter 5. Exchange matrix construction is discussed in Chapter 6.

Chapter 5

Coulomb Matrix Construction

The Coulomb matrix is essential for all SCF–type quantum chemistry calculations: it is needed in HF as well as in KS–DFT, for both pure and hybrid functionals. This chapter focuses on how multipole approximations can be used to accelerate the computation of the Coulomb matrix. The Coulomb matrix $\mathbf{J}$ is given by

$$J_{pq} = \sum_{rs} D_{rs}(pq|rs) \tag{5.1}$$

Taking into account the expression (2.20) for the two-electron integrals $(pq|rs)$, this can be rewritten as

$$J_{pq} = \int \frac{b_p(\mathbf{r_1})b_q(\mathbf{r_1}) \sum_{rs} D_{rs} b_r(\mathbf{r_2})b_s(\mathbf{r_2})}{|\mathbf{r_1} - \mathbf{r_2}|} d\mathbf{r_1} d\mathbf{r_2} = \int \frac{b_p(\mathbf{r_1})b_q(\mathbf{r_1})\rho(\mathbf{r_2})}{|\mathbf{r_1} - \mathbf{r_2}|} d\mathbf{r_1} d\mathbf{r_2} \tag{5.2}$$

where $\rho(\mathbf{r})$ is the electronic density

$$\rho(\mathbf{r}) = \sum_{rs} D_{rs} b_r(\mathbf{r})b_s(\mathbf{r}) \tag{5.3}$$

This means that the Coulomb matrix element J_{pq} can be interpreted as follows: *J_{pq} is the Coulombic repulsion energy of an electron whose spatial distribution is given by $b_p(\mathbf{r})b_q(\mathbf{r})$, due to the total charge distribution $\rho(\mathbf{r})$ of all electrons.*

To efficiently evaluate Coulomb matrix elements J_{pq} it is useful to have two things precomputed. Firstly, a list of non-negligible basis function products $b_p(\mathbf{r})b_q(\mathbf{r})$, as described in section 4.4. Secondly, a description of the density $\rho(\mathbf{r})$ that allows one to evaluate repulsion from a group of charges that are far away, without considering interactions of individual pairs of charges. Such a description of $\rho(\mathbf{r})$ is used in the fast multipole method, discussed in the next section.

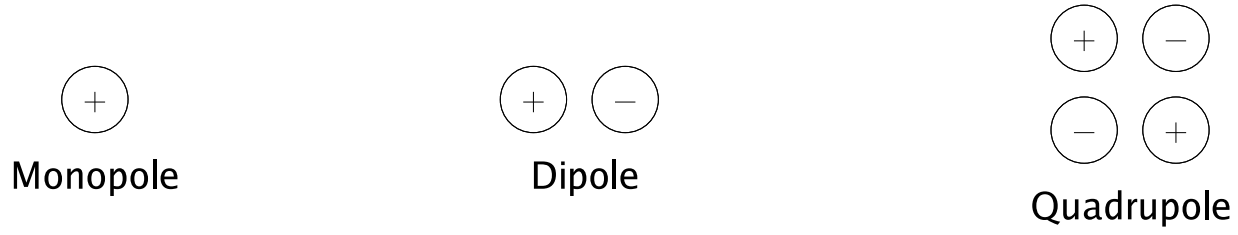
5.1 The Fast Multipole Method

The Fast Multipole Method (FMM) was originally developed for calculation of the interaction between classical point charges[9, 10]. Later, much research has been devoted to the application of multipole approximations in quantum chemistry [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. The basic idea of FMM is that a group of charges that are close to each other can be described by a multipole expansion, which includes enough information to accurately evaluate the Coulomb repulsion between that group of charges and some other charge that is well separated from the group. This allows Coulomb interaction to be calculated without taking into account each individual charge separately, thus reducing the computational effort. A hierarchy of groups of charges is used, so that larger groups of charges can be used at long distances. This gives improved scaling.

When FMM is applied in quantum chemistry with GT–LCAO basis functions, the "charges" are no longer point charges but instead Gaussian charge distributions arising from products of basis functions. Groups of such charge distributions are then described by multipole expansions.

So what is a "multipole expansion"? It is a list of numbers called "multipole moments", each multipole moment giving the weight of one particular contribution to the expansion. The simplest part is the monopole contribution. Then there are three dipole contributions, one for each coordinate direction. Higher order contributions correspond to more complicated multipoles; quadrupoles, octopoles, etc. The expansion must be truncated at some expansion order, chosen so that the desired accuracy is maintained.

Figure 5.1: Multipole moments.



Each Coulomb matrix element J_{pq} is computed as a sum of a near–field (NF) part and a far–field (FF) part:

$$J_{pq} = J_{pq}^{\mathrm{NF}} + J_{pq}^{\mathrm{FF}} \tag{5.4}$$

The near–field part J_{pq}^{NF} is computed by explicit evaluation of the integrals $(pq|rs)$, while

the far–field part J_{pq}^{FF} is computed as

$$J_{pq}^{\mathrm{FF}} = \sum_i \int \frac{b_p(\mathbf{r_1}) b_q(\mathbf{r_1}) \rho_i(\mathbf{r_2})}{|\mathbf{r_1} - \mathbf{r_2}|} d\mathbf{r_1} d\mathbf{r_2} \tag{5.5}$$

where the sum is taken over all groups of charges, each with electron density $\rho_i(\mathbf{r})$. Each term in the sum (5.5) is computed using a truncated multipole expansion of the density $\rho_i(\mathbf{r})$:

$$\int \frac{b_p(\mathbf{r_1}) b_q(\mathbf{r_1}) \rho_i(\mathbf{r_2})}{|\mathbf{r_1} - \mathbf{r_2}|} d\mathbf{r_1} d\mathbf{r_2} \approx \mathbf{q}_{pq}^T \mathbf{T} \mathbf{q}_i \tag{5.6}$$

Here, $\mathbf{T}$ is the multipole interaction matrix, and $\mathbf{q}_{pq}$ and $\mathbf{q}_i$ are the multipole expansions (i.e. vectors of multipole moments) for the basis function pair pq and the density ρ_i respectively.

Using (5.6), the interaction between the basis function pair pq and a large part of the electron density can be evaluated at once, saving lots of work compared to the alternative of explicitly computing all needed integrals $(pq|rs)$. By using a hierarchy of spatial boxes, one can make sure that larger groups of charges are used at longer distances, see Figure 5.2.

To compute the whole Coulomb matrix $\mathbf{J}$, a large number of interaction of the type (5.6) must be evaluated, and each vector of multipole moments $\mathbf{q}$ can be re–used many times. The multipole moments (components of the vector $\mathbf{q}$) describing a charge distribution $\rho(\mathbf{r})$ are computed as

$$q_k = \int \rho(\mathbf{r}) f_k(\mathbf{r} - \mathbf{r}_c) \mathrm{d}\mathbf{r} \tag{5.7}$$

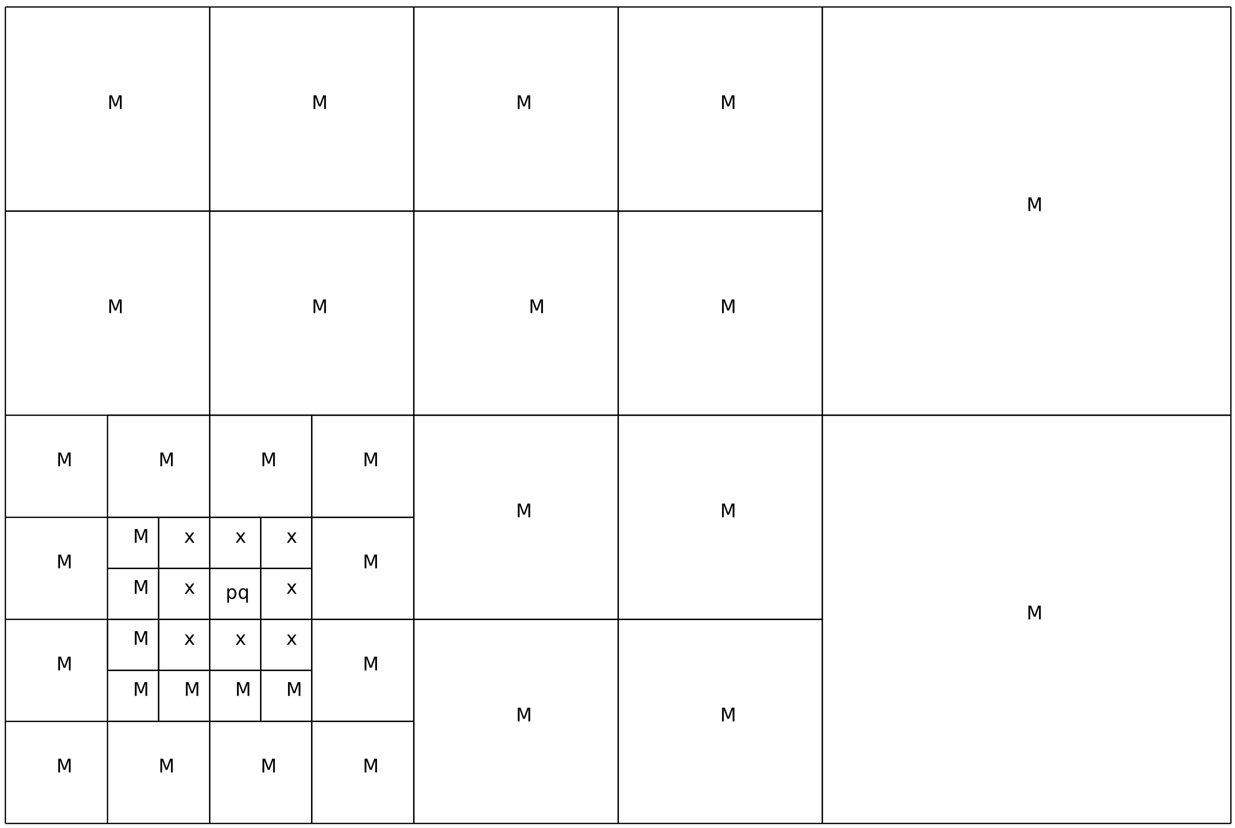
where $\mathbf{r}_c$ is the chosen center point for the multipole expansion. The functions $f_k(\mathbf{r})$ are the real regular solid harmonics[7, 21]. The functions of zeroth, first, and second order are as follows:

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl} & & & & & & f_5(\mathbf{r}) & = & \frac{1}{2}\sqrt{3}(x^2 - y^2) \\ & & & f_2(\mathbf{r}) & = & x & f_6(\mathbf{r}) & = & \sqrt{3}xz \\ f_1(\mathbf{r}) & = & 1 & f_3(\mathbf{r}) & = & y & f_7(\mathbf{r}) & = & \frac{1}{2}(3z^2 - r^2) \\ & & & f_4(\mathbf{r}) & = & z & f_8(\mathbf{r}) & = & \sqrt{3}yz \\ & & & & & & f_9(\mathbf{r}) & = & \sqrt{3}xy \end{array} \tag{5.8}$$

Thus, the first multipole moment is simply the total charge: $q_0 = \int \rho(\mathbf{r}) \mathrm{d}\mathbf{r}$. Higher order moments involve polynomials of increasing degrees.

In general, infinitely high order of multipole moments is needed for an exact description of a charge distribution $\rho(\mathbf{r})$. However, when evaluating Coulomb interaction at long distances, the higher order moments become less important as the distance increases. At very long distances, the monopole contribution (total charge) is the only thing that matters. At intermediate distances, the interaction is well described using a truncated multipole

Figure 5.2: Schematic picture of the hierarchy of boxes used to compute the Coulomb matrix element J_{pq}. Multipole descriptions of the electron density are used for boxes marked with M, while interaction with boxes marked with X are treated by explicit evaluation of the corresponding two–electron integrals $(pq|rs)$.



expansion, where only the lower–order moments are included. This is the reason why truncated multipole expansions are so useful for Coulomb interactions: by computing only the lower–order multipole moments q_k, we are extracting the most relevant information about the charge distribution $\rho(\mathbf{r})$.

The multipole interaction matrix $\mathbf{T}$ in (5.6) depends on the separation vector between the interacting multipoles[7]. Therefore, $\mathbf{T}$ must be re–evaluated many times. Both the computation of $\mathbf{T}$ and the subsequent contraction with multipole moments (5.6) become very time–consuming if a high order of multipole moments is used. Therefore, much efficiency can be gained by choosing as low multipole expansion order as possible for each interaction. On the other hand, when truncated multipole expansions are used to describe the electron density, this approximation introduces errors. The most important issue discussed in Paper 1

of this thesis is how the order of multipole expansion can be selected dynamically, depending on the needed accuracy in each part of the Coulomb matrix computation.

Chapter 6

Exchange Matrix Construction

The HF exchange matrix is needed in HF and in hybrid KS–DFT. The exchange matrix $\mathbf{K}$ is given by

$$K_{pq} = \sum_{rs} D_{rs}(pr|qs) \tag{6.1}$$

Note the difference in the order of indexes compared to the Coulomb matrix. There has been much research devoted to efficient computation of the exchange matrix[23, 24, 25, 26, 27, 28, 29, 30]. In particular, the so–called LinK algorithm [26, 28] has been very successful and is now used in several quantum chemistry codes. This chapter gives a brief discussion on how and why linear scaling can be achieved in exchange matrix construction.

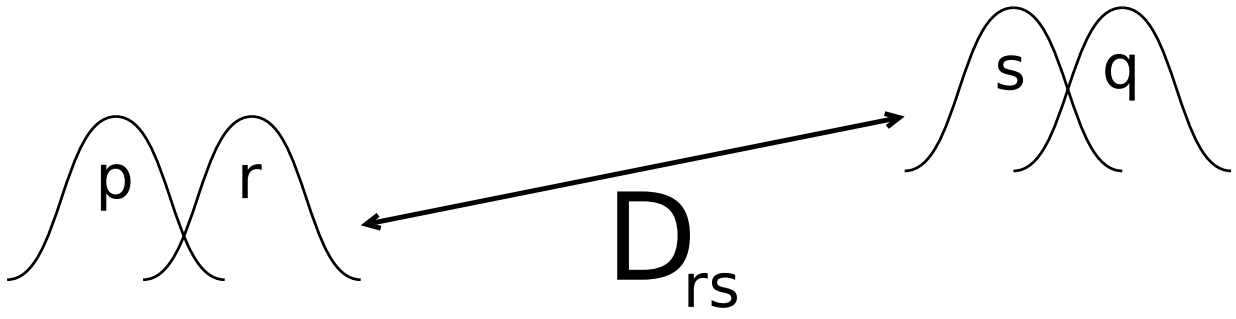
With the assumption that any density matrix element D_{rs} is zero when the corresponding basis functions are far enough apart, and recalling the expression (2.20) for the two-electron integrals $(pq|rs)$, one can understand why linear scaling is possible for the exchange by considering the contribution $D_{rs}(pr|qs)$. For such a contribution to be nonzero, the following conditions must be met (see Figure 6.1):

- The basis functions $b_p(\mathbf{r})$ and $b_r(\mathbf{r})$ must be close enough to each other so that the product $b_p(\mathbf{r})b_r(\mathbf{r})$ is nonzero.
- The basis functions $b_q(\mathbf{r})$ and $b_s(\mathbf{r})$ must be close enough to each other so that the product $b_q(\mathbf{r})b_s(\mathbf{r})$ is nonzero.
- The basis functions $b_r(\mathbf{r})$ and $b_s(\mathbf{r})$ must be close enough to each other so that the density matrix element D_{rs} is nonzero.

But then all four basis functions b_p, b_q, b_r, and b_s must be close to each other! So, for a particular density matrix element D_{rs}, the number of nonzero exchange matrix contributions

$D_{rs}(pr|qs)$ that must be evaluated is constant (independent of system size) for large enough systems. Since the number of nonzero density matrix elements scales linearly with system size, the number of nonzero contributions $D_{rs}(pr|qs)$ needed for the construction of the exchange matrix $\mathbf{K}$ also scales linearly with system size.

Figure 6.1: Schematic illustration of how the basis function products $b_p(\mathbf{r})b_r(\mathbf{r})$ and $b_q(\mathbf{r})b_s(\mathbf{r})$ are coupled through the density matrix element D_{rs}.



In order to compute the exchange matrix $\mathbf{K}$ in a linear scaling fashion, one must predict the size of contributions $D_{rs}(pr|qs)$ without explicitly computing them. Also, one must be able to skip many contributions at once, without performing work to estimate the size of each individual contribution. This can be done by precomputing the basis function products $b_p(\mathbf{r})b_q(\mathbf{r})$ and sorting them according to their size as measured by the Cauchy–Schwarz measure $\sqrt{(pq|pq)}$. Given such a sorted list of basis function products, it is possible to skip many exchange matrix contributions at a time. This is one of the key ideas used in the LinK algorithm[26, 28].

Chapter 7

Density Matrix Purification

An essential part of the self–consistent field procedure, described in section 2.4.2, is the construction of a new density matrix for a given Fock matrix $\mathbf{F}$. This is traditionally done by first solving the generalized eigenvalue problem

$$\mathbf{FC} = \mathbf{SC}\boldsymbol{\Lambda} \tag{7.1}$$

The operation of solving (7.1) to find $\mathbf{C}$ and $\boldsymbol{\Lambda}$ is referred to as *diagonalization* of $\mathbf{F}$. Then, the new density matrix $\mathbf{P}$ is constructed from the eigenvectors of $\mathbf{F}$ as

$$P_{pq} = \sum_{a=1}^{n_{occ}} C_{pa} C_{qa} \tag{7.2}$$

The expression (7.2) differs from (2.24) by a factor of 2. This is only for convenience, rescaling the entire matrix by a scalar is of course easily done.

For large systems, constructing the new density matrix according to (7.1) and (7.2) becomes very time–consuming since the diagonalization operation scales as $\mathcal{O}(n^3)$. Density matrix purification[31, 32, 33, 34, 35, 36, 37, 38, 39, 40] provides a way to compute the density matrix without explicit knowledge of the orbitals, thus avoiding the expensive diagonalization step. This section is intended to give an introduction to the concept of density matrix purification, giving some background for Paper 7 of this thesis.

7.1 Transformation to Standard Form

Before considering density matrix purification, we will transform the eigenvalue problem (7.1) to standard form, using a matrix $\mathbf{Z}$ such that $\mathbf{Z}^T\mathbf{Z} = \mathbf{S}^{-1}$. With the congruence

transformation $\widehat{\mathbf{F}} = \mathbf{Z}^T\mathbf{F}\mathbf{Z}$ the eigenvalue problem is then written

$$\widehat{\mathbf{F}}\widehat{\mathbf{C}} = \widehat{\mathbf{C}}\mathbf{\Lambda} \tag{7.3}$$

The resulting eigenvectors $\widehat{\mathbf{C}}$ are related to $\mathbf{C}$ as $\mathbf{C} = \mathbf{Z}\widehat{\mathbf{C}}$, so from

$$\widehat{P}_{pq} = \sum_{a=1}^{n_{occ}} \widehat{C}_{pa}\widehat{C}_{qa} \tag{7.4}$$

we can recover the density matrix $\mathbf{P} = \mathbf{Z}\widehat{\mathbf{P}}\mathbf{Z}^T$. Density matrix purification provides a way of computing $\widehat{\mathbf{P}}$ for a given $\widehat{\mathbf{F}}$. The complete procedure of computing $\mathbf{P}$ for a given $\mathbf{F}$, including the congruence transformation, is as follows:

$$\begin{aligned} \widehat{\mathbf{F}} &= \mathbf{Z}^T\mathbf{F}\mathbf{Z} \\ \widehat{\mathbf{D}} &= \text{Puri}(\widehat{\mathbf{F}}) \\ \mathbf{P} &= \mathbf{Z}\widehat{\mathbf{P}}\mathbf{Z}^T \end{aligned}$$

The matrix $\mathbf{Z}$ may be chosen in many different ways. One option is to use an inverse Cholesky decomposition of $\mathbf{S}$ as discussed in Paper 2 of this thesis.

7.2 Purification

We will now consider how density matrix purification can be used to compute $\widehat{\mathbf{P}}$ for a given $\widehat{\mathbf{F}}$. It follows from (7.3) and (7.4) that the density matrix $\widehat{\mathbf{P}}$ has the following properties:

- Any eigenvector of $\widehat{\mathbf{F}}$ is also an eigenvector of $\widehat{\mathbf{P}}$.
- The eigenvalues of $\widehat{\mathbf{P}}$ are either 0 or 1. Thus, $\widehat{\mathbf{P}}$ is *idempotent*: $\widehat{\mathbf{P}}^2 = \widehat{\mathbf{P}}$.
- The n_{occ} eigenvectors of $\widehat{\mathbf{F}}$ that correspond to the smallest eigenvalues are eigenvectors of $\widehat{\mathbf{P}}$ with eigenvalue 1, while the other eigenvectors of $\widehat{\mathbf{P}}$ have eigenvalue 0.

Another way of saying this is that the matrices $\widehat{\mathbf{F}}$ and $\widehat{\mathbf{P}}$ differ only in their eigenvalues: if one takes the matrix $\widehat{\mathbf{F}}$ and moves the n_{occ} smallest eigenvalues to 1 and all other eigenvalues to 0, one ends up with the density matrix $\widehat{\mathbf{P}}$.

But is there any way to manipulate the eigenvalues of a matrix without first explicitly computing all eigenvalues and eigenvectors? The answer is yes!

Consider a matrix $\mathbf{A}$ with an eigenvector $\mathbf{x}$ and corresponding eigenvalue λ:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{7.5}$$

We get the following for the square of the matrix $\mathbf{A}$:

$$\mathbf{A}^2\mathbf{x} = \mathbf{A}\mathbf{A}\mathbf{x} = \mathbf{A}\lambda\mathbf{x} = \lambda\mathbf{A}\mathbf{x} = \lambda^2\mathbf{x} \tag{7.6}$$

Therefore, if $\mathbf{x}$ is an eigenvector of $\mathbf{A}$ with eigenvalue λ, then $\mathbf{x}$ is also an eigenvector of $\mathbf{A}^2$ with eigenvalue λ^2. In a similar way one can see that if $\mathbf{x}$ is an eigenvector of $\mathbf{A}$ with eigenvalue λ_A and at the same time an eigenvector of $\mathbf{B}$ with eigenvalue λ_B, then

$$(\mathbf{A} + \mathbf{B})\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{x} = (\lambda_A + \lambda_B)\mathbf{x} \tag{7.7}$$

It follows from (7.6) and (7.7) that any polynomial function of a matrix will act only on the eigenvalues of the matrix, while the eigenvectors remain the same. In our case we are interested in moving the eigenvalues of $\widehat{\mathbf{F}}$ in a certain way; to achieve that we can apply a cleverly chosen polynomial function to $\widehat{\mathbf{F}}$. Technically, this means we need to carry out a series of matrix–matrix multiplications and additions. This is the basic idea of density matrix purification. If the matrices are sparse enough, both matrix–matrix multiplication and addition can scale linearly with system size. Provided that the number of matrix operations needed is independent of system size, this gives a linearly scaling way of constructing the density matrix.

So, when using density matrix purification, we start with the matrix $\widehat{\mathbf{F}}$, then perform a series of matrix manipulations until we end up with the corresponding density matrix $\widehat{\mathbf{D}}$. The most time–critical matrix operation used is the evaluation of the square of the matrix. In order to carry out the matrix operations efficiently, we must make sure the matrices are sparse throughout the purification procedure. Unfortunately, when a matrix square $\mathbf{B} = \mathbf{A}^2$ is computed, the resulting matrix $\mathbf{B}$ is usually less sparse than than the original matrix $\mathbf{A}$. In order to keep the matrices sparse, one must remove some small (hopefully negligible) matrix elements. It is not obvious how this removal of small elements from the matrices will affect the accuracy of the final result $\widehat{\mathbf{D}}$. Paper 7 of this thesis addresses this problem of error control in density matrix purification; it gives a recipe for how small matrix elements can be removed in a systematic way while ensuring that the requested accuracy is still reached in the final result.

Chapter 8

Storage and Manipulation of Matrices

This chapter contains a brief discussion about how matrices occurring in SCF calculations are stored and handled. A more thorough discussion on this subject can be found in Emanuel Rubensson's licentiate thesis[41].

All matrices involved in Hartree–Fock or Kohn–Sham calculations need to be stored in a way suitable for the kind of matrix operations one needs to perform on them. Typical needed matrix operations are matrix–matrix multiply, addition of matrices, and computation of the trace of a matrix. Some more complicated operations are also of interest, such as diagonalization and inverse Cholesky decomposition.

When treating small systems, full matrix storage is the natural choice; a $n \times n$ matrix A is then stored as a row– or column–wise list of its elements, so that a matrix element A_{ij} is accessed as $A[i*n+j]$ or as $A[j*n+i]$. Full matrix storage is convenient because most matrix operations needed can be done using highly optimized linear algebra software libraries. For large systems, however, full matrix storage is not optimal because many matrix elements are negligibly small, so that they do not need to be stored. Much can then be gained by exploiting sparsity in the matrices.

8.1 Sparsity

The matrices occurring in HF or KS-DFT calculations with GT–LCAO basis functions are such that each matrix element A_{ij} corresponds to a pair of basis functions $b_i(\mathbf{r})$ and $b_j(\mathbf{r})$. If the two basis functions are non–overlapping and separated by a large distance, the corresponding matrix element is usually of small magnitude. For this reason, it is advantageous to order the basis functions in such a way that basis functions that are near each other in space are also near each other in the ordering. Then, negligible matrix elements

will often be located together, so that a *block–sparse* matrix storage can be used. Block–sparse matrix storage is much preferable to element–wise sparse matrix storage because highly optimized linear algebra software libraries can be used at the level of submatrices.

8.2 Hierarchic Representation

Different types of block–sparse matrix storage are possible. One commonly used option is to use the Blocked Compressed Sparse Row (BCSR) format, in which only the non–zero submatrices of each row of the matrix are stored. The column indexes of the non–zero submatrices are kept in a separate vector. Another option is to use a *hierarchic* matrix data structure, in which the simple full–matrix storage is used at each level. This approach has certain advantages over the BCSR format. Paper 2 of this thesis describes our implementation of a hierarchic matrix data structure, which we refer to as the Hierarchic Matrix Library (HML). As an example of what is gained compared to the BCSR format, Paper 2 of this thesis includes a description of how the inverse Cholesky decomposition can be performed using HML.

Chapter 9

Large Quantum Chemistry Calculations in Practice

This chapter discusses some aspects of large–scale quantum chemistry calculations that are seldom mentioned when results are reported in scientific publications, although they can be very important in practice. Most of the issues discussed below come from experiences from working with the HF/DFT program *Ergo*[42] in which the algorithms discussed in previous chapters have been implemented, using the C++ programming language.

9.1 Hardware

The type of computer hardware available is of course of great importance for large scale quantum chemistry calculations. I worked with computer resources at three different computing centers:

- PDC: Center for Parallel Computers, at KTH in Stockholm.
- HPC2N: High Performance Computing Center North, in Umeå.
- NSC: National Supercomputer Centre, in Linköping.

The Lenngren cluster at PDC and the Sarek cluster at HPC2N are similar in that they both have 8 GB of memory and two processors per node. Most of my calculations were run on those two systems. The Mozart machine at NSC has 64 processors and 512 GB of shared memory, thus making really large calculations possible. However, there are many users who compete for CPU time on Mozart, and therefore this machine has been difficult to use in practice. For example, from the moment when you decide to run a calculation on Mozart,

there may be a delay of a month or so before the calculation actually starts. Such delays may mean that your project cannot be finished within reasonable time.

9.2 Compilers

There are often several different compilers on the computer system where you want to run your calculations. Ideally, you should choose the compiler that gives the best performance. For example, for Intel processors there is usually an Intel compiler available, which is supposed to produce executable machine code specifically optimized for that processor and thus significantly faster than the code generated by the free GNU C/C++ compiler (gcc) which is available on almost all machines.

Before I started working with these things, I was under the illusion that a compiler was something robust and predictable. Something you could trust, like the ground you stand on as a programmer. Oh, sweet innocence! Later I have become very much aware of the fact that a compiler is just another piece of software, as full of bugs as any other program. During the last two years' work on the *Ergo* program, we have encountered bugs in all compilers used: Intel, Portland Group, Pathscale, gcc (or maybe not gcc, not sure about that one). And not only once for each of them, but several times new bugs appeared in new versions of the compilers.

A compiler bug is something quite painful for a programmer. First, you somehow notice that your program does not work as it should. It takes a while to verify that this is really so, and not just something wrong in the input. Then, when you are sure something is really wrong, you start tracking the error down. This may be difficult when the program consists of many different parts, different source files, written by different people. At this point you probably think there is a bug in your own code (after all, compilers are supposed to work properly). Then, after hours of work on isolating the problem, you end up with a small test program, usually just 10–20 lines of code, that reproduces the error. It then becomes evident that there was nothing wrong with your code in the first place, but it is the compiler that is buggy. The solution is to report the bug by sending the small test program to someone who can hopefully do something about it. In the meantime, you have to use another compiler.

We did not count the hours of work spent on finding and reporting compiler bugs, but there has been a lot of that. In retrospect, I think it would have been better to simply stick to the gcc compiler, which is by far more reliable than any of the others, rather than wasting so much time on finding bugs in compilers just to make the program run a little bit faster.

One example: in April 2007 we reported a bug in the Intel compiler version 9.1. After a long time the Intel support team replied that they will not fix the bug in version 9.1 but that we should wait for version 10 which should arrive soon. When version 10.0 arrived, it contained new bugs, which we reported. Version 10.1 was just released, but the bug reported for 10.0 is still there in 10.1. We are still waiting for a version that works. To summarize: for more than six months there has been no version of the Intel compiler available which could compile our code correctly, despite the fact that we spent much work on finding and reporting the bugs.

9.3 Maintenance and Documentation of Code

A quantum chemistry program is rather complex, with many different parts that must work together. Therefore, it is important to design and organize the code in a way that makes it easy to understand and modify. This is even more important if many people are involved in the development.

Using a version control tool is one way of making it easier to handle coding projects. For the *Ergo* source code, the version control tool Subversion (svn) was used. This tool helps keeping track of all changes made in the code, with information about who made each change, when the change was done, and hopefully a note briefly explaining what was done. It also makes it possible to retrieve a copy of the source code as it looked at a previous time, which is most helpful in case you suspect that a bug has been introduced.

Another thing to think about is how to organize the program into independent modules as much as possible, with a well–defined and documented set of tasks that each module is responsible for. This kind of program design has many advantages:

- It becomes easier to understand how the program works.
- New functionality can be added without rewriting too much of the existing code.
- Separate unit tests can be written, each of which tests the functionality of only a single module, making it easier to find errors.
- Modules give a natural way of dividing work and responsibilities between different people.

Documentation is another critical issue. Whenever code is written or modified, this should be accompanied by some comments in the source code explaining what is going on.

Source code without any comments can be really difficult to maintain. However, there is one thing even worse than uncommented code: code with *obsolete, no longer valid comments*! It is very important for developers to be disciplined in this respect.

The programming language is also important. The *Ergo* code is written in C++, which has some features such as strict type control and class inheritance which help in creating a sound program design. One should really try to use any means possible to make the code more understandable, and also get as much help as possible finding errors already at the compilation stage. In C++, an example of this is the use of the `const` keyword, which tells the compiler to make sure that a variable that is not supposed to be changed really remains constant. If the programmer by mistake writes code that would manipulate that variable, this will result in a compilation error, so that the mistake can be corrected immediately.

9.4 Convergence and Starting Guesses

There is no guarantee that the SCF procedure as described in Section 2.4.2 will converge; this depends on the quality of the starting guess density. If the starting guess is far away from an energy minimum, one may need to modify the procedure somewhat in order to converge. In the *Ergo* program this is done by taking smaller steps, as follows: When a new Fock matrix $\mathbf{F}_{\text{new}}$ has been computed, it is mixed with the previous Fock matrix $\mathbf{F}_{\text{prev}}$ as

$$\mathbf{F} = \alpha\mathbf{F}_{\text{new}} + (1-\alpha)\mathbf{F}_{\text{prev}} \tag{9.1}$$

where α is a parameter between 0 and 1 such that $\alpha = 1$ corresponds to the standard SCF procedure, while a smaller value of α gives a more "careful" procedure with smaller steps.

Direct inversion in the iterative subspace (DIIS)[43, 44] is a very useful method for accelerating convergence in SCF calculations. Using DIIS can reduce the number of SCF iterations significantly provided that the current electron density is close enough to the final solution. The *Ergo* code uses mixing with the previous Fock matrix according to (9.1) in early iterations if the starting guess is bad, and switches to the DIIS scheme for the final iterations.

In the HF and KS–DFT calculations discussed in this thesis, the starting guess electron density was usually taken from a previous cheaper calculation in which larger threshold values and/or a smaller basis set was used. In order to create a new starting guess density matrix using basis set A with n_A basis functions from a density matrix computed using a different basis set B with n_B basis functions, we need the overlap matrix between the two basis sets $\mathbf{R}$:

$$R_{pq} = \int b_p^A(\mathbf{r}) b_q^B(\mathbf{r}) d\mathbf{r} \tag{9.2}$$

Note that if basis sets A and B are the same, $\mathbf{R}$ is nothing but the ordinary overlap matrix $\mathbf{S}$. Given a density matrix $\mathbf{D}_A$ computed using basis set A, we get the corresponding projected density matrix for basis set B by applying the usual purification procedure to the matrix $-\mathbf{R}^T\mathbf{D}_A\mathbf{R}$:

$$\mathbf{D}_B = \mathrm{Puri}(-\mathbf{R}^T\mathbf{D}_A\mathbf{R}) \tag{9.3}$$

where Puri() stands for the purification operation including the congruence transformation. If $n_A \neq n_B$ the matrix $\mathbf{R}$ will be rectangular, but the product $\mathbf{R}^T\mathbf{D}_A\mathbf{R}$ is again quadratic and symmetric, with dimension n_B.

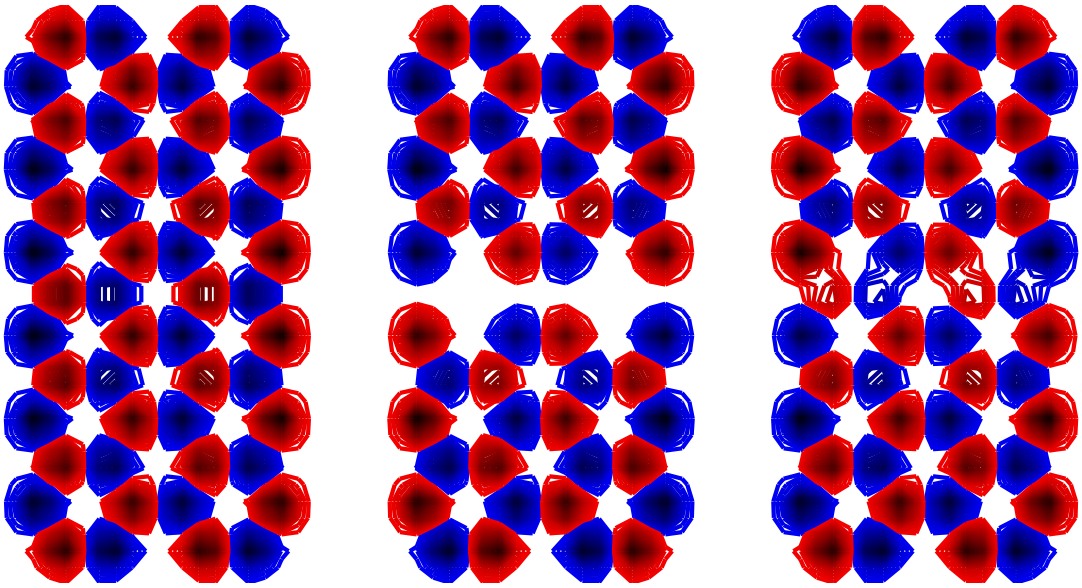
This procedure of using a projection of a previously computed density as starting guess can save much computational time since the time and memory requirements for a calculation depend very much on the chosen threshold values and basis set, so that an initial low–accuracy calculation to generate the starting guess can be done at an almost negligible cost compared to the following high–accuracy calculation. Of course, the very first calculation for a particular system must be done without any previous density to start from. Then, one can simply set the starting guess density matrix to a diagonal matrix. Many iterations may then be needed to converge, but the calculation is still cheap compared to the desired final calculation.

In some cases a minimal basis set such as STO-2G can be used in an initial calculation to provide a starting guess density. Sometimes I even used a "STO-1G" basis set to obtain starting guesses for STO-2G calculations. One word of warning about minimal basis sets, though: there are cases when a calculation using a minimal basis set will not converge at all. In particular, for large protein–like systems calculations with STO-XG type of basis sets will not converge due to a vanishing HOMO–LUMO gap. Then, a somewhat larger basis set must be used to generate the starting guess. The 3-21G basis set seems to be good enough to give a reasonable HOMO–LUMO gap in such cases. To speed up initial calculations, we sometimes used a hybrid STO-2G/3-21G basis set, where STO-2G basis functions are used for Carbon and Hydrogen atoms, and 3-21G basis functions are used for all other atoms.

Note that this procedure of taking a density from a previous calculation as starting guess is not strictly necessary; it merely saves time. For example, if a final result for the 6-31G** basis set is desired, one could run a single 6-31G** calculation using a diagonal matrix as starting guess. However, such a bad guess means that many extra iterations are needed to converge. Therefore, a more efficient way of getting the same result is to run an initial calculation using the 3-21G basis set and take that density as starting guess for the 6-31G** calculation.

The starting guess is not only important for the time it takes to complete the calculation, but may also affect the result in case there are several local energy minima. It may happen

Figure 9.1: Spin density plots for three Hartree–Fock calculations on a small graphene ribbon. These three calculations had identical input except for the starting guesses, which were taken as diagonal matrices with different random distortions. The energy of the leftmost configuration is considerably lower than the other two.



that different calculations using the same method and basis set reach different results due to different starting guesses. If the ground state energy is desired, one must then compare the final energies of the different calculations to see which one is the lowest. Of course, it is also possible that the ground state was not found by any of the calculations, since the desired ground state corresponds to a global minimum on a potential surface that may have many local minima, and the optimization procedure used is designed to find a local minimum. Therefore, if there is any doubt that the computed result really corresponds to the ground state, it is good practice to verify that the same result can be reproduced using a different starting guess.

The choice of starting guess was important in the calculations of Paper 3. For that kind of systems, many local minima exist in which the spin density pattern is not optimal. See Figure 9.1 for an example of this.

9.5 Disk Usage

In theoretical method development one often refers to the resource requirements of a method in terms of two quantities: the number of operations (e.g. CPU time) and the needed storage space. The type of storage space is not considered. In practice, however, it is of great importance if things can be stored in computer RAM memory rather than on the hard-drive, since memory access is typically at least 1000 times faster than disk access.

9.6 Reproducibility

When working with method development, one is always interested in comparing the own implementation to others. It is then very important that results are reproducible. Therefore, it is good practice to always keep molecular geometries used so that they are available to anyone who wishes to reproduce the results. Also, reporting numbers such as total energies and band gaps makes it easier to reproduce results.

Chapter 10

Final Remarks

10.1 Comments on Included Papers

- Paper 1: *Efficient implementation of the fast multipole method*
  In this article, we describe our implementation of multipole approximations. Several aspects are discussed, of which the most important is the use of dynamically chosen multipole expansion order. Although a similar approach had been employed already in 1995 by Challacombe and coworkers [15, 16], our formulation is different and in my opinion easier to understand.

- Paper 2: *A hierarchic sparse matrix data structure for large-scale Hartree-Fock/Kohn-Sham calculations*
  This paper describes the Hierarchic Matrix Library, which is extensively used for sparse matrices in the *Ergo* program, in particular for inverse Cholesky decomposition and for density matrix purification. Using the hierarchic representation of sparse matrices, we have been able to implement all necessary matrix operations so that full matrix storage is no longer used in any part of the code.

- Paper 3: *Nonlocal exchange interaction removes half-metallicity in graphene nanoribbons*
  This study of graphene nanoribbons was inspired by the work of Son et al[45, 47], whose DFT calculations on graphene nanoribbons showed an interesting behavior of the spin density. We could reproduce most of their results, but found some differences since we used a different DFT functional. During the work we also found discrepancies regarding the magnitude of spin density, which we reported to Son et al. This turned out to be due to a bug in their software, which led to errata in Nature[46] and in Physical Review Letters[48].

- Paper 4: *A linear scaling study of solvent-solute interaction energy of drug molecules in aqua solution*
  In this work we used HF and DFT/B3LYP calculations to calculate the interaction energy to water for three different drug molecules. Our main conclusion was that in this kind of calculations it is necessary to include water molecules in a rather large domain around the solute molecule; it is not enough to consider only a few of the nearest water molecules.

- Paper 5: *Basis Set Dependence of Solute-Solvent Interaction Energy of Benzene in Water: A HF/DFT study*
  This was a continuation of the work on interaction energy to water, this time focusing on the basis set dependence and basis set superposition error[49]. Both Paper 4 and Paper 5 deal with methodology to compute solute–solvent interaction energies, which are important for determining the solubility of the compound. However, our calculations were only done for a few configurations of water molecules around the solute. In order to get reliable information about the solute–solvent interaction energy, many more configurations would be needed. Also, one should consider the molecular dynamics force field used to generate the configurations. One could hope that the chosen force field has only a small effect on the computed solute–solvent interaction energy. Otherwise, ab–initio molecular dynamics simulations may be needed to generate reliable solute–solvent interaction energies. More details about solubility problems can be found in Laban Bondesson's doctoral thesis[50].

- Paper 6: *Rotations of occupied invariant subspaces in self–consistent field calculations*
  In this paper we describe how errors in the electron density in SCF calculations can be seen as rotations of the subspace spanned by all occupied eigenvectors of the Fock and density matrices. This way of measuring errors makes it possible to determine what accuracy is needed for different approximations used in the calculation. This is very useful since in large calculations computational approximations are absolutely necessary; without any approximations the calculation would take far too long time to be practical. Originally, the idea of considering the occupied subspace was something that Emanuel came up with while working on density matrix purification. Therefore, Paper 6 and Paper 7 were first written as a single manuscript. Then we decided it was better to separate them, so that Paper 6 is general and applies to approximations in both Fock matrix construction and density matrix construction.

- Paper 7: *Density matrix purification with rigorous error control*
  In our first implementations of density matrix purification in 2005, the matrix was truncated (that is, small elements were removed) in the same way in each step of

the purification procedure. Then, the error compared to the exact case without any truncation behaved in a way we could not quite understand: in early iterations the error increased a lot, while in the later iterations the error increased very little. This was really a mystery at that time. Now the mystery is solved, as explained in Paper 7. Looking at the paper now, it is funny to note that things are written in a very different order compared to how they actually happened. For example, the theorems on how to relate eigenvalues of a near–idempotent matrix $\mathbf{X}$ to the norm $\|\mathbf{X} - \mathbf{X}^2\|$ and the traces of $\mathbf{X}$ and $\mathbf{X}^2$ were among the first things we did, but have ended up as a side note in the very end of the article.

- Paper 8: *Estimation of errors in Coulomb and exchange matrix construction*
  In this work we numerically investigated how the Euclidean norm of the error matrix for approximately computed Coulomb and exchange matrices depends on the threshold value τ, see section 4.3. We found a simple relationship, which means that the threshold τ can be chosen so that a requested accuracy in the resulting Coulomb or exchange matrix is obtained, as measured by the Euclidean norm of the error matrix. Together with Paper 7, this means that both Fock matrix construction and the density matrix construction can be done with controlled error in the Euclidean norm, so that the scheme for error control described in Paper 6 can be applied in practice.

- Paper 9: *Hartree-Fock calculations with linearly scaling memory usage*
  In this paper we report timings and memory usage for Hartree–Fock calculations on test systems of different sizes, with emphasis on the scaling behavior. The point is to show that we have managed to combine linear scaling methods for Coulomb and exchange matrix construction, matrix operations, and density matrix purification in our program in such a way that all parts of the code scale linearly with system size, with respect to both time and memory requirements. Test calculations were run for system sizes as large as possible on the type of computer used, which had 32 GB of memory shared by four processors. The largest calculation we managed to run on that kind of machine was a Hartree–Fock calculation on a glutamine–alanine helix with 11650 atoms, corresponding to 67204 basis functions using the 3-21G basis set.

- Paper 10: *Truncation of small matrix elements based on the Euclidean norm for blocked data structures*
  In this note, we describe a procedure for removal of as many small matrix elements as possible from a matrix while making sure that the Euclidean norm of the resulting error matrix stays below a requested threshold. This truncation method is used in our implementation of density matrix purification, where the Euclidean norm based truncation allows us to keep track of eigenvalue movement as described in Paper 7.

10.2 The Publication Process

This section is not specific to large quantum chemistry calculations, but contains a few comments about the review process used in scientific journals, which applies to almost anyone working with research.

Research results are usually published in so–called *peer reviewed* journals. This means that the editor of the journal consults one or more experts in the field in order to decide whether or not a submitted article will be accepted for publication in the journal, in an attempt to ensure that the articles published are of good quality and relevant to the field.

From the author's point of view, the procedure is as follows. You choose a journal in which you want to publish your paper. After submitting your manuscript to the journal, it is understood that you are not allowed to submit it for publication elsewhere until the journal has made a decision. After a few weeks, you get a reply from the journal with some comments from one or several reviewers. Depending on the reviewer's opinion, the manuscript may be accepted for publication, or it may be rejected, or you may be asked to change it and submit a revised version.

Provided that the editors do their job, this procedure works rather nicely. Unfortunately, this is not always the case. Some journals have a habit of keeping you waiting several months for a review. I have encountered at least two such journals:

- Journal of Computational Physics (J. Comput. Phys.) We submitted a manuscript to J. Comput. Phys. on May 12, 2006. The review answer came nearly five months later, on October 4.
- Physical Review B (Phys. Rev. B) After submitting a manuscript to Phys. Rev. B, we had to wait nearly four months for the review.

In both cases we got impatient and sent emails to the editors asking them what was taking so long, but from their replies it was clear that they did not consider 4-5 months to be a particularly long time for a review. Those two journals, Journal of Computational Physics and Physical Review B, are now on my black–list. I find it completely amazing that they can get away with delays of 4–5 months in the review process. Such delays have many negative consequences:

- Other researchers in the field do not get access to the new results.
- The authors are hindered in publishing their continued work where they probably need to refer to the delayed manuscript.

- Even if the authors were first with the presented results, someone else manages to publish similar results while the original manuscript is still under review.

The overall consequence is that the research field as a whole suffers; scientific progress is slowed down because the latest research results are not published directly but are instead kept hidden away in a desk drawer of some journal's editorial office.

Fortunately, not all journals are as slow as those on the black–list above. An example of a journal that has a reasonable policy for review time is the Journal of Chemical Physics (J. Chem. Phys.). When submitting a manuscript to J. Chem. Phys., you can expect the review in three weeks. This does not mean that the demands are lower; as far as I can tell, papers published in J. Chem. Phys. are of high quality, no worse than any of the two amazingly slow journals mentioned above.

To summarize: when you want to publish your research results, you should carefully choose which journal you submit to. Beware that while some journals will arrange for a review in a few weeks time, others may sit on your manuscript for half a year! It would be interesting to know why they do that. I really don't know.

10.3 Future Outlook

In this final section are some speculations on what might happen in the future regarding method development in the field of quantum mechanics for large systems. Of course, like anyone trying to forsee the future, I am only guessing!

10.3.1 Error Control

The scheme for error control in SCF calculations described in Paper 6 should be implemented so that the threshold values used in each part of the calculation are automatically determined by the accuracy in the current trial density. Then, instead of the current situation with several different threshold values for different parts of the code, only a single threshold value will be needed for each calculation, namely the requested convergence of the occupied subspace. This should make it much easier to see where the bottlenecks in the program really are, something that has been difficult so far because of the different threshold values.

Another challenge is to investigate how error control can be achieved in the exchange–correlation matrix in KS–DFT calculations. Possibly a similar approach as in Paper 8 can be used.

10.3.2 Parallelization

Parallelization of the code is necessary to make good use of modern computer resources. Adapting existing algorithms to a distributed memory architecture will require some work, but really needs to be done since the majority of the new supercomputer facilities are of the distributed memory type.

10.3.3 Direct Computation of Energy Differences

Using methods like those described in this thesis, it is possible to achieve linear scaling with system size for calculations at a certain level of accuracy. Unfortunately, for a given choice of threshold values for the calculations, errors in total energies also increase linearly with system size. When energy differences are computed by explicit subtraction of total energies from calculations on large systems, as in Paper 4 and Paper 5, the error in the resulting energy difference grows with the system size even if the energy difference itself remains small. A possible remedy for this is direct computation of energy differences, as described in Ref. [51]. I think such approaches will become increasingly important in the future.

10.3.4 More Accurate Methods, Beyond HF and KS–DFT

In many cases, the HF and DFT methods are not accurate enough. There are more accurate methods available, such as the Møller-Plesset perturbation theory (MP) and coupled–cluster (CC) methods. These more accurate methods are much more time–consuming than HF/DFT, and the scaling behavior makes it difficult to apply such methods to large systems. However, there are promising results indicating that the scaling of the MP and CC methods can be improved[52, 53, 54].

Methods based on the two–electron reduced density matrix (2–RDM)[55, 56, 57, 58] are interesting as alternatives to wave function based methods such as MP and CC. In the 2–RDM methods, the two–electron density matrix is optimized directly; knowledge about the actual wave function in terms of Slater determinants is not needed. So far, 2–RDM methods have only been applicable to very small systems, but it may be possible to improve the scaling by exploiting sparsity in the matrices, in the same way as for HF and KS–DFT. The two–electron density matrix is a four–index object, so without sparsity its storage scales as $\mathcal{O}(n^4)$. However, if elements corresponding to well–separated basis functions can be neglected, the scaling behavior may be improved. Sparsity in four–dimensional arrays has been exploited in the context of MP methods[52]; something similar might be possible

for 2–RDM methods. An important advantage of 2–RDM methods compared to MP and CC is that no reference wave function is needed.

Bibliography

[1] P.A.M. Dirac, Proceedings of the Royal Society A123, 714 (1929).

[2] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*, Dover, N. Y., 1996.

[3] W. Kohn and L.J. Sham, Phys. Rev., **140**:A1133, 1965.

[4] P. Hohenberg and W. Kohn, Phys. Rev., **136**:B864, 1964.

[5] A. D. Becke, J. Chem. Phys **98**, 5648 (1993).

[6] F. Abu-Awwad and P. Politzer, J. Comput. Chem. **21**, 227 (2000).

[7] T. Helgaker, P. Jørgensen and J. Olsen, *Molecular Electronic-Structure Theory*, Wiley, Chichester, 2000.

[8] M. Häser and R. Ahlrichs, *Improvements on the direct SCF method*, J. Comput. Chem. **10**, 104 (1989).

[9] L. Greengard and V. Rokhlin, J. Comput. Phys. **73**, 325 (1987).

[10] K. E. Schmidt and M. A. Lee, J. Stat. Phys. **63**, 1223 (1991).

[11] I. Panas, J. Almlöf, and M. W. Feyereisen, Int. J. Quantum Chem. **40**, 797 (1991).

[12] I. Panas and J. Almlöf, Int. J. Quantum Chem. **42**, 1073 (1992).

[13] C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon, Chem. Phys. Lett. **230**, 8 (1994).

[14] C. A. White and M. Head-Gordon, J. Chem. Phys **101**, 6593 (1994).

[15] M. Challacombe, E. Schwegler, and J. Almlöf, J. Chem. Phys **104**, 4685 (1995).

[16] M. Challacombe and E. Schwegler, J. Chem. Phys **106**, 5526 (1997).

[17] C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon, Chem. Phys. Lett. **253**, 268 (1996).

[18] M. C. Strain, G. E. Scuseria, and M. J. Frisch, Science **271**, 51 (1996).

[19] C. H. Choi, K. Ruedenberg, and M. S. Gordon, J. Comput. Chem. **22**, 1484 (2001).

[20] M. Sierka, A. Hogekamp, and R. Ahlrichs, J. Chem. Phys. **118**, 9136 (2003).

[21] M. A. Watson, P. Sałek, P. Macak, and T. Helgaker, J. Chem. Phys. **121**, 2915 (2004).

[22] C. K. Gan, C. J. Tymczak, and M. Challacombe, J. Chem. Phys **121**, 6608 (2004).

[23] E. Schwegler and M. Challacombe, J. Chem. Phys. **105**, 2726 (1996).

[24] J. C. Burant and G. E. Scuseria, J. Chem. Phys. **105**, 8969 (1996).

[25] E. Schwegler, M. Challacombe, and M. Head-Gordon, *Linear scaling computation of the Fock matrix. II. Rigorous bounds on exchange integrals and incremental Fock build*, J. Chem. Phys. **106**, 9708 (1997).

[26] C. Ochsenfeld, C. A. White and M. Head-Gordon, J. Chem. Phys. **109**, 1663 (1998).

[27] E. Schwegler and M. Challacombe, J. Chem. Phys. **111**, 6223 (1999).

[28] C. Ochsenfeld, Chem. Phys. Lett. **327**, 216 (2000).

[29] D. S. Lambrecht and C. Ochsenfeld, J. Chem. Phys. **123**, 184101 (2005).

[30] F. Aquilante, T. B. Pedersen, and R. Lindh, J. Chem. Phys. **126**, 194106 (2007).

[31] R. McWeeny, Proc. R. Soc. London Ser. A **235**, 496 (1956).

[32] C. Ochsenfeld and M. Head-Gordon, Chem. Phys. Lett. **270**, 399 (1997).

[33] A. H. R. Palser and D. E. Manolopoulos, Phys. Rev. B **58**, 12704 (1998).

[34] A. M. N. Niklasson, Phys. Rev. B **66**, 155115 (2002).

[35] A. M. N. Niklasson, C. J. Tymczak, and M. Challacombe, J. Chem. Phys. **118**, 8611 (2003).

[36] A. Holas, Chem. Phys. Lett. **340**, 552 (2001).

[37] D. A. Mazziotti, Phys. Rev. E **68**, 066701 (2003).

[38] R. Pino and G. E. Scuseria, Chem. Phys. Lett. **360**, 117 (2002).

[39] A. M. N. Niklasson, C. J. Tymczak and H. Røder, Phys. Rev. B **66**, 155120 (2002).

[40] H. J. Xiang, W. Z. Liang, J. Yang, J. G. Hou, and Q. Zhu, J. Chem. Phys. **123**, 124105 (2005).

[41] E. H. Rubensson, *Sparse Matrices in Self-Consistent Field Methods*, Licentiate thesis, Theoretical Chemistry, KTH, Stockholm (2006).

[42] E. Rudberg, E. H. Rubensson, and P. Sałek, *Ergo version 1.6*, a quantum chemistry program for large-scale self-consistent field calculations, 2007.

[43] P. Pulay, Chem. Phys. Lett. **73**, 393 (1980).

[44] P. Pulay, J. Comput. Chem. **3**, 556 (1982).

[45] Y. W. Son, M. L. Cohen, and S. G. Louie, Nature **444**, 347 (2006).

[46] Y. W. Son, M. L. Cohen, and S. G. Louie, Nature **446**, 342 (2007).

[47] Y. W. Son, M. L. Cohen, and S. G. Louie, Phys. Rev. Lett. **97**, 216803 (2006).

[48] Y. W. Son, M. L. Cohen, and S. G. Louie, Phys. Rev. Lett. **98**, 089901 (2007).

[49] F. B. van Duijneveldt, J. G. C. M. van Duijneveldt-van de Rijdt, and J. H. van Lenthe, Chem. Rev. **94**, 1873 (1994).

[50] L. Bondesson, *Microscopic Interpretations of Drug Solubility*, Doctoral thesis, Theoretical Chemistry, KTH, Stockholm (2007).

[51] A. Beste, R. J. Harrison, and T. Yanai, J. Chem. Phys. **125**, 074101 (2006).

[52] V. Weijo, P. Manninen, P. Jørgensen, O. Christiansen, and J. Olsen, J. Chem. Phys **127**, 074106 (2007).

[53] J. E. Subotnik and M. Head-Gordon, J. Chem. Phys. **122**, 034109 (2005).

[54] P. E. Maslen, A. D. Dutoi, M. S. Lee, Y. H. Shao, and M. Head-Gordon, Mol. Phys. **103**, 425 (2005).

[55] D. A. Mazziotti, Phys. Rev. Lett. **97**, 143002 (2006).

[56] D. A. Mazziotti, Phys. Rev. A **74**, 032501 (2006).

[57] D. A. Mazziotti, Phys. Rev. A **75**, 022505 (2007).

[58] D. A. Mazziotti, J. Chem. Phys **126**, 184101 (2007).