Thesis: implementations of core computational chemistry algorithms

Andrey Asadchev

Contents

Chapter 1

Introduction

The primary goal of computational chemistry is of course to predict chemical properties: energy, gradients, Hessians (vibrational frequencies), and other properties for a given chemical system. For example, to find the excitation energy or rotation barriers one would perform a series of single point energy calculations. To find local extrema on the potential energy surface a series of energy gradient calculations are needed, and so on.

The computational science aspect of computational chemistry is often treated as a necessary evil. Over the years, it so happened that most of the designers and authors of quantum chemistry algorithms and their implementations were chemists and physicists first, and computational scientists second.

The foundations for the quantum chemistry were developed before the World War II. For example the Hartree-Fock [1, 2] method was developed in the late 20's, and the foundation of perturbation theory [3] dates back to the mid-30's. However, practical application of the theoretical methods did not come until the emergence of sufficient computing resources to crunch the numbers.

20th century scientific computing was dominated by Fortran, short for Formula Translator, one of the earliest programming languages, first developed in the 50's [4]. The computers and operating systems at the inception of Fortran were expensive proprietary products, batch machines running stacks of manually prepared inputs. Compared to today's powerful computers, computing in the 50's and the 60's may as well have been done on clay tablets.

In the 70's another language, C [5], and a new operating system, UNIX, came out of Bell Labs. With the rise of UNIX, the C programming language gained strong footing among computer science and computer engineering practitioners. In the same decade Cray produced its first groundbreaking supercomputer, Cray I, which gave researchers for the first ability to crack tough numerical problems, such as weather prediction, in a timely manner. In the field of computational chemistry many of the core programs (some still in use today) were developed and incorporated into computational chemistry packages, notably HONDO [6] and GAUSSIAN [7]. A majority of this work was spearheaded by John Pople, who won the 1998 Nobel Prize for his contribution to the field.

The 80's saw the growth of UNIX and standardization of system interfaces with POSIX and SystemV [8] standards. C++ [9], a multi-paradigm language based on C, was being developed by Bjarne Stroustrup at Bell Labs. To avoid limitations placed on software by patents and restrictive licensing, Richard Stallman began the GNU foundation, which sought to liberate software development. The GNU Compilers Collection (GCC) and GNU public licenses are perhaps the most visible of the many contributions GNU made to computing and scientific fields. The decade also witnessed the birth of massively parallel supercomputers, such as the Thinking Machines. To take advantage of the emerging trends in scientific computing, a number of parallel computational chemistry algorithms were developed, including parallel Hartree-Fock [10] and MP2 [11]. In the early 80's Purvis and Bartlett first implemented a coupled cluster singles and doubles algorithm [12], or CCSD for short. Subsequently, CCSD with perturbative triples correction method [13], CCSD(T), was developed which today is the gold standard of computational chemistry. In the same decade, GAMESS [14] began to be developed, with HONDO as much of its initial codebase.

In the 90's, the exotic supercomputers of the previous decades slowly disappeared, starved from generous military budgets of the Cold War which was now over [15, 16]. The burgeoning personal computer market funneled billions of dollars into research and development of commodity Intel and AMD processors. The fragmented UNIX market was slowly eroded by the ever maturing Microsoft Windows and a new operating system, Linux. Started as a hobby in the early 90's by Linus Torvalds, Linux, released under GNU Public License, quickly caught the interest of programmers worldwide and within a few years became one the major operating systems of the Internet age. The C++ programming language became the preferred choice for writing complex applications, albeit not just yet in the scientific fields. However, more and more scientific codes of the 90's were run and developed for clusters of commodity computers running Linux and connected by relatively inexpensive networks. One of the more interesting developments in computational chemistry was NWChem [17], a set of codes designed specifically with parallel distributed memory systems in mind. NWChem was perhaps the last major computational chemistry package whose development started primarily in Fortran.

The Internet bubble burst at the turn of the 21st century, spelling financial problems and consequent death to the many flagship companies of the last century, including SUN and SGI. With the release of X86-64 extensions by AMD in 2003, commodity processors became a full-fledged 64-bit architecture, suitable for any computational challenge. By the 2010's the processor market became dominated almost exclusively by multicore AMD and Intel chips, with IBM still retaining some presence in the high-end computing market with its Power processors. The latest development in the commodity computing is the reemergence of accelerators, such as using graphics cards to solve general programs, so-called General Processing on GPU (GPGPU). The leader in the field has been NVIDIA with its CUDA [18] technology, but recently Intel joined the market with its Many Integrated Cores (MIC) technology [18]. The efforts to unify development across regular microprocessors and various accelerators led to OpenCL [18], a set open standards for developing applications that run across heterogeneous platforms.

The software development in scientific communities has steadily shifted towards C/C++.

While there is still a lot of legacy code written in Fortran (and hence continuing development), much of the new development happens in C++ and Python [19]. Examples are Q-Chem [20], with most of its new development happening in C++, and Psi4 [21], almost entirely implemented in C++ with Python used as a scripting engine. The C++ language and compilers continue to evolve and improve at a faster pace than Fortran, mostly due to influence of much larger commercial application development market. In terms of raw speed, the C++ programs are as fast as Fortran counterparts but C++ has the advantage of modern programming techniques and many libraries and frameworks, e.g. Boost [22].

So, what does the contemporary scientific computing platform look like now? It is almost always a distributed memory cluster of very fast multicore computers, with between 2 and 64 GB of memory per node. Some clusters might have GPU accelerators to augment the computational power. The number of cores in the cluster varies greatly, from just a few to tens of thousands. The interconnect can be 1Gb Ethernet, an InfiniBand, of proprietary network, such as SeaStar on Cray supercomputers. The file system can be a local disk or a parallel file system capable of storing terabytes of data.

Ultimately, it is the hardware (or rather the hardware limitations) that dictates how the algorithm is to be designed. Until we have infinite memory and bandwidth, the algorithms will always have to be designed with these limitations in mind. Furthermore, the algorithms have to be designed so as to account for a great variety of system configurations. Few general rules of thumb can be used as general guidelines for designing scalable and efficient algorithms: minimize communication, keep memory footprint low and introduce adjustable parameters for memory use, use external libraries, e.g. Linear Algebra Package [23] (LAPACK), and make software easy to modify, extend, and even rewrite, perhaps by using certain programming language over another. Furthermore, how will the scientific computing landscape look in the future? Who knows! But we better design the software so that changes dictated by the hardware can be accommodated efficiently.

In the following chapters are attempts to develop a modern, but simple and flexible, C++

foundation for computational chemistry algorithms and several algorithm implementations built upon that foundation with the above rules of thumb in mind.

But before one can get into the intertwined details of science, algorithms, and hardware some theoretical background is necessary to explain to the reader in broad detail the basis sets, two-electron integrals, and transformations which will form the bulk of the subsequent pages.

1.1 Hartree-Fock

At the center of computational chemistry is the evaluation of the time-independent Schrdinger equation eigenvalue problem,

$$H\Psi = E\Psi$$

where H is the Hamiltonian operator, Ψ is the wavefunction containing all the relevant information about the chemical system, E is the energy of the system and eigenvalue of the Hamiltonian. To be a proper wavefunction, Ψ must be square integrable and normalized, $\langle\Psi|\Psi\rangle = 1$, and antisymmetric to satisfy the Pauli exclusion requirement for fermions. The expectation value E then can be computed as:

$$\langle\Psi|H|\Psi\rangle = E$$

In terms of individual contributions, the Schrodinger equation can be written in terms of the kinetic and potential energies of the electrons and nuclei:

$$(T_e + T_n + V_{ee} + V_{en} + V_{nn})\Psi = E\Psi$$

T_e and T_n are the kinetic energy terms for electrons and nuclei respectively

$$T_e = -\sum_{e}^{N_e} \frac{\nabla_e^2}{2}$$

$$T_n = -\sum_{n}^{N_n} \frac{\nabla_n^2}{2m_n}$$

∇^2 is the Laplacian operator,

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$

V_{ee} is the electron-electron repulsion term,

$$V_{ee} = \sum_{e<f}^{N_e} \frac{1}{r_{ef}}$$

V_{en} is the electron-nucleus attraction term,

$$V_{ee} = -\sum_{e}^{N_e} \sum_{n}^{N_n} \frac{Z_n}{r_{en}}$$

V_{nn} is the nucleus-nucleus repulsion term,

$$V_{nn} = \sum_{n<l}^{N_n} \frac{Z_n Z_l}{r_{nl}}$$

The closed form analytic solution for the Schrodinger equation exists only for the simplest systems, such as those with one or two particles. To evaluate a quantum system of interest, a number of approximations has to be made. In the Born-Oppenheimer approximation [24] the much slower nuclei are treated as stationary point charges and the Schrodinger equation then reduces to the electronic Schrodinger equation:

$$H_e = T_e + V_{ee} + V_{en}$$

$$H_e\Psi = E_e\Psi$$

The general problem of the type $\langle\Psi|\frac{1}{r_{ij}}|\Psi\rangle$ has no analytic solution and further approximations must be made. The crudest solution is to assume that electrons do not interact with each other. This leads to the independent particle model in which,

$$\Psi_{IPM} = \phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2)...$$

is separable with respect to each electron coordinate vector.

The independent particle wavefunction does not satisfy the anti- symmetry requirement, but properties of the determinant do (since exchanging any two rows or columns changes the sign). Taking the determinant of Ψ_{IPM} leads to Slater determinant Ψ_{HF} which in turn leads to the Hartree-Fock method

$$\Psi_{HF}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(\mathbf{x}_1) & \psi_2(\mathbf{x}_1) & \cdots & \psi_N(\mathbf{x}_1) \\ \psi_1(\mathbf{x}_2) & \psi_2(\mathbf{x}_2) & \cdots & \psi_N(\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ \psi_1(\mathbf{x}_N) & \psi_2(\mathbf{x}_N) & \cdots & \psi_N(\mathbf{x}_N) \end{vmatrix}$$

After performing a series of algebraic manipulations, the closed-shell Hartree-Fock energy can be written as

$$E_{HF} = \langle\Psi_{HF}|H_e|\Psi_{HF}\rangle = 2\sum_i h_{ii} + \sum_{ij}(2J_{ij} - K_{ij})$$

where h_{ii} is the one-electron integral

$$h_{ij} = (\psi_i|h_1|\psi_j) = \int \psi_i(-\frac{\nabla^2}{2} - \sum_n^{N_n} \frac{Z_n}{r_n})\psi_j dr_1$$

and J and K terms, called Coulomb and exchange, respectively, are two-electron integrals

$$J = (ij|ij)$$

$$K = (ij|ji)$$

$$(ij|kl) = \int\int \psi_i(r_1)\psi_j(r_1)\frac{1}{r_{12}}\psi_k(r_2)\psi_l(r_2)dr_1dr_2$$

From now on, the HF label will be dropped and Ψ will be understood to refer to Ψ_{HF} and H to refer H_e.

The only constraint on the one particle orbitals is that they remain orthonormal,

$$\langle\psi_i|\psi_j\rangle = \delta_{ij}$$

Therefore the orbitals can be manipulated to affect energy. According to the variational principle, the best orbitals are those that minimize the energy,

$$\partial E = \frac{\partial\Psi}{\partial\psi} = 0$$

The method of Lagrange multipliers solves minimization problem with constraints. The resulting Lagrange equation

$$\partial[\langle\Psi|H|\Psi\rangle - \sum_{i,j}(\lambda_{ij}\langle\psi_i|\psi_j\rangle - \delta_{ij})] = 0$$

can be reduced to

$$F\psi_k = \lambda_{ij}\psi_k$$

where F is the Fock operator

$$F = [h_1 + \sum_i(2J_i - K_i)]$$

Taking the Lagrangian multipliers to be of the form

$$\lambda_{ij} = \delta_{ij}\epsilon_k$$

the Hartree-Fock minization problem becomes an eigenvalue problem:

$$F\psi_k = \epsilon_k \psi_k$$

Optimizing general orbitals in the above problem is not generally feasible. Instead Roothaan [25] proposed to expand orbitals in terms of a known basis and restrict optimization to coefficients:

$$\psi_i = \sum_b^N c_{ib} \chi_b$$

The optimization of molecular orbitals ψ_i in terms of a fixed basis leads to the Hartree-Fock-Roothaan equations

$$FC = \epsilon SC$$

where C is the coefficient matrix and $S = (\chi_i|\chi_j)$ is the basis overlap matrix. The above equation is almost a solvable eigenvalue equation, except for the S term. Although a general basis is not usually orthonormal, it can be orthonormalized in which case the overlap matrix becomes the identity matrix, $S = \delta_{ij}$ and the Hartree-Fock-Roothaan equation takes the form of a regular eigenvalue problem

$$FC = \epsilon C$$

Now an expression for the Fock operator can be derived in terms of the coefficients and one- and two- electron integrals over basis functions

$$F = (\chi_i|h_1|\chi_j) + D[(\chi_i\chi_j|h_2|\chi_k\chi_l) - (\chi_j\chi_k|h_2|\chi_j\chi_l)]$$

where $D = 2\sum_i c_{ib} * c_{ib}$ is known as density matrix.

Since the orbital coefficients appear on both sides of the equation, the Hartree-Fock method needs to be repeated until the difference between the old and the new coefficients reaches a certain threshold. Because of that, the Hartree-Fock method is also called the self consistent field (SCF) method.

The simple interpretation of the Hartree-Fock method is that an electron is moving in the mean field of the other electrons. The interaction of individual electrons is not correlated, other than accounting for the Pauli exclusion principle. Accounting for electronic interaction will be discussed below.

1.2 Basis Set

To understand the intricate details of the computational chemistry algorithms, especially when discussing two-electron integrals, a few words must be said about the basis set.

Modern basis sets are based on the atomic orbitals, which are spatial orbitals reminiscent of the $s, p, ...$ orbital shapes found in physical chemistry books. Because of that the basis sets are often called atomic basis or atomic orbitals, as opposed to molecular orbitals, which are simply atomic orbitals transformed via the coefficient matrix C.

The correct shape for an (Cartesian) atomic orbital is the Slater-type orbital (STO)

$$Ax^l y^m z^n e^{-\alpha r}$$

where A is the normalization coefficient and l, m, n are related to the angular momentum quantum number L,

$$L = l + m + n$$

Using a Gaussian function, a similar type of orbital, called Gaussian-type orbital (GTO), can be devised

$$Ax^l y^m z^n e^{-\alpha r^2}$$

Unlike the Gaussian functions, the Slater functions cannot be separated into x, y, z components, making the evaluation of integrals over the Slater basis expensive. On the other hand, Gaussian function can be written as

$$e^{-\alpha r^2} = e^{-\alpha x^2} e^{-\alpha y^2} e^{-\alpha z^2}$$

and due to this property, the computation of integrals over the Gaussian functions is much simpler [26], with a number of different closed-form solutions for one- and two- electron integrals [27, 28, 29, 30]. Most electronic structure programs use GTOs as basis sets. An exception to this trend is Amsterdam Density Functional (ADF) program suite [31] which uses STOs.

To reproduce the approximate shape of an STO, a linear combination of several GTOs can be taken and fitted according to some criteria, a process known as contraction and the resulting orbital called contracted Gaussian-type orbital,

$$\chi_{cgto} = Ax^l y^m z^n \sum_{k}^{K} C_k e^{-\alpha_k r^2}$$

where K is the construction order and C_k are the contraction coefficients. In this context, the individual Gaussians are called primitives.

The individual contracted orbitals which share the same primitives are grouped together into shells. The primary reason for doing so is computational efficiency. With a correct algorithm, only the angular term $x^l y^m z^n$ will be different between shell functions; the terms involving expensive exponent computations will be the same.

The simplest contracted basis sets are of the STO-NG family [32], where N is the number of contracted GTOs fitted to an STO using a least-squares method. The major difference between GTOs and STOs is the function shape near the origin, where GTOs are flat and STOs have a cusp. This is especially important for the core electrons near the nucleus. More advanced basis sets typically have more GTOs to represent contracted core orbitals (6-10 GTO) and fewer GTOs to represent non-core orbitals (1-3 GTOs). This segmented approach strikes a delicate balance between accuracy and computational time.

It should be obvious that a larger basis set will give better orbitals and lower energy, based on the Variational Principle. However, larger basis set will also increase computational time, may lead to slower convergence, and may result in numerical instabilities. A majority of time is spent evaluating two-electron integrals and building the Fock matrix. Although, atomic integrals do not change from iteration to iteration, storing N^4 elements can be prohibitively expensive for any large system, and thus the integrals can be re-computed on-the-fly. Currently, Hartree-Fock computations with a few thousand basis functions are routinely performed in a matter of hours. In the near future that number is likely to be the tens of thousands.

1.3 Electron Correlation

As a rule of thumb, the energy computed with the Hartree-Fock method accounts for $\tilde{9}9$ % of the total electronic energy. However, the physical properties associated with the last 1 % of energy are what usually is sought. Hartree-Fock computations can give very good geometries, but the energy differences can only be qualitative at best.

Recall from the above discussion that Hartree-Fock model does not account for instantaneous electronic interaction, but instead treats each electron as interacting with an electronic mean field. The difference between the total energy and the Hartree-Fock energy is called the correlation energy

$$E_{corr} = E_{hf} - E$$

To recover the correlation energy, Hartree-Fock computations must be followed by what are called correlation methods, which try to recover the correlation energy from the Hartree-Fock wavefunction. In the context of electron correlation computations, Hartree-Fock is typically the zeroth order (also called the reference) wavefunction. Among the many correlation methods there are two that are central to the next chapters: the MP2 and coupled cluster.

The formula for the MP2 energy is relatively simple, expressed only in terms of molecular integrals $(ia|jb)$ and orbital energies ϵ

$$E_{MP2} = \sum_{ij} \sum_{ab} \frac{[2(ai|bj) - (bi|aj)](ai|bj)}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}$$

As is customary in many-body methods, indices $i, j, ...$ refer to occupied molecular orbitals O, $a, b, ...$ to virtual orbitals V, and p, q, r, s to atomic basis N.

The time consuming part of the MP2 energy computation is not the actual energy computation, which scales as O^2N^2, but the transformation from atomic to molecular integrals (also called 4-index transformation), which scale as ON^4. Another bottleneck in many-body methods is the storage of molecular integrals. For a large MP2 calculation the storage may well be on the order of terabytes. The details of MP2 energy computation will be covered in detail in the corresponding chapter.

The coupled cluster theory was first proposed in nuclear physics [33] and later adopted in quantum chemistry by Cizek [34] as the exponential ansatz

$$\Psi = e^T \Psi_0 = e^{(T_1 + T_2 + ... T_n)} \Psi_0$$

where $T_1 ... T_n$ are the n-particle excitation operator and Ψ_0 is the reference wavefunction, typically Ψ_{hf} in computational chemistry. The excitation operator applied to a reference wavefunction is written in terms of excitation amplitudes t from hole states $i, j, k, ...$ (also referred to as occupied orbitals) to particle states $a, b, c, ...$ (also referred to as virtual orbitals),

$$T_n \Psi_0 = \sum_{ijk...} \sum_{abc...} t_{ijk...}^{abc...} \Psi_{ijk...}^{abc...}$$

The CCSD algorithm is an iterative process that scales as $N^2V^2O^2$ and the triples correction (T) scales as N^2V^4O. To compute the CCSD(T) energy, every type of four-index molecular integral is needed. The coupled cluster algorithm will be covered in detail in the last chapter. Both, MP2 and CC can be easily and systematically derived using Goldstone diagrams, a diagrammatic approach to nonrelativistic fermion interaction based on Feynman diagrams. A very thorough treatment of the many-body theory can be found in the excellent book by Shavitt and Bartlet [35].

Bibliography

[1] D. R. Hartree. *Proc. Cambridge Phil. Soc.*, 24:89, 1928.

[2] V. Fock. Nherungsmethode zur lsung des quantenmechanischen mehrkrperproblems. *Zeitschrift fr Physik A Hadrons and Nuclei*, 61:126–148, 1930. 10.1007/BF01340294.

[3] C. Møller and M.S. Plesset. Note on an approximation treatment for many-electron systems. *Physical Review*, 46(7):618, 1934.

[4] J.W. Backus, R.J. Beeber, S. Best, R. Goldberg, L.M. Haibt, H.L. Herrick, R.A. Nelson, D. Sayre, P.B. Sheridan, H. Stern, et al. The fortran automatic coding system. In *Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability*, pages 188–198. ACM, 1957.

[5] B.W. Kernighan and D.M. Ritchie. *The C programming language*. Prentice Hall, 1988.

[6] Dupuis. M, Rys. J, and King H. F. Hondo. Quantum Chemistry Program Exchange, 11, 336338, 1977.

[7] W. J. Hehre, W. A. Lathan, R. Ditchfield, M. D. Newton, , and J. A. Pople. Gaussian 70. Quantum Chemistry Program Exchange, Program No. 237, 1970.

[8] J. Isaak. Standards-the history of posix: a study in the standards process. *Computer*, 23(7):89–92, 1990.

[9] B. Stroustrup. *The C++ programming language*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[10] M. Dupuis and JD Watts. Parallel computation of molecular energy gradients on the loosely coupled array of processors (lcap). *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)*, 71(2):91–103, 1987.

[11] J.D. Watts and M. Dupuis. Parallel computation of the moller–plesset second-order contribution to the electronic correlation energy. *Journal of computational chemistry*, 9(2):158–170, 1988.

[12] G.D. Purvis III and R.J. Bartlett. A full coupled-cluster singles and doubles model: The inclusion of disconnected triples. *The Journal of Chemical Physics*, 76:1910, 1982.

[13] K. Raghavachari, G.W. Trucks, J.A. Pople, and M. Head-Gordon. A fifth-order perturbation comparison of electron correlation theories. *Chemical Physics Letters*, 157(6):479–483, 1989.

[14] M. S. Gordon and M. W. Schmidt. *Advances in electronic structure theory: GAMESS a decade later*, pages 1167–1189. Elsevier, Amsterdam, 2005.

[15] Cold war's end hits cray computer. *New York Times*, 1992.

[16] In supercomputers, bigger and faster means trouble. *New York Times*, 1994.

[17] DE Bernholdt, E. Apra, HA Früchtl, MF Guest, RJ Harrison, RA Kendall, RA Kutteh, X. Long, JB Nicholas, JA Nichols, et al. Parallel computational chemistry made easier: The development of nwchem. *International Journal of Quantum Chemistry*, 56(S29):475–483, 1995.

[18] A. Heinecke, M. Klemm, and H. Bungartz. From gpgpu to many-core: Nvidia fermi and intel many integrated core architecture. *Computing in Science & Engineering*, 14(2):78–83, 2012.

[19] Python. http://python.org/.

[20] Y. Shao, L.F. Molnar, Y. Jung, J. Kussmann, C. Ochsenfeld, S.T. Brown, A.T.B. Gilbert, L.V. Slipchenko, S.V. Levchenko, D.P. ONeill, et al. Advances in methods and algorithms in a modern quantum chemistry program package. *Phys. Chem. Chem. Phys.*, 8(27):3172–3191, 2006.

[21] J. M Turney, A. C Simmonett, R. M Parrish, E. G Hohenstein, F. Evangelista, J.T. Fermann, B.J. Mintz, L.A. Burns, J.J. Wilke, M. L Abrams, et al. Psi4: an open-source ab initio electronic structure program. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2011.

[22] Boost c++ libraries. http://www.boost.org/.

[23] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, et al. Lapack usersguide: Release 1.0. Technical report, Argonne National Lab., IL (United States), 1992.

[24] M. Born and R. Oppenheimer. Zur quantentheorie der molekeln. *Annalen der Physik*, 389(20):457–484, 1927.

[25] C.C.J. Roothaan. New developments in molecular orbital theory. *Reviews of modern physics*, 23(2):69, 1951.

[26] SF Boys. Electronic wave functions. i. a general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 200(1063):542–554, 1950.

[27] J.A. Pople and W.J. Hehre. Computation of electron repulsion integrals involving contracted gaussian basis functions. *Journal of Computational Physics*, 27(2):161–168, 1978.

[28] J. Rys, M. Dupuis, and H. F. King. Computation of electron repulsion integrals using the rys quadrature method. *Journal of Computational Chemistry*, 4(2):154–157, 1983.

[29] S. Obara and A. Saika. Efficient recursive computation of molecular integrals over cartesian gaussian functions. *The Journal of chemical physics*, 84:3963, 1986.

[30] M. Head-Gordon and J.A. Pople. A method for two-electron gaussian integral and integral derivative evaluation using recurrence relations. *The Journal of chemical physics*, 89:5777, 1988.

[31] G. Te Velde, F.M. Bickelhaupt, E.J. Baerends, C. Fonseca Guerra, S.J.A. van Gisbergen, J.G. Snijders, and T. Ziegler. Chemistry with adf. *Journal of Computational Chemistry*, 22(9):931–967, 2001.

[32] J.A. Pople. Molecular orbital methods in organic chemistry. *Accounts of Chemical Research*, 3(7):217–223, 1970.

[33] F. Coester and H. Kümmel. Short-range correlations in nuclear wave functions. *Nuclear Physics*, 17:477–485, 1960.

[34] J. Čížek. On the correlation problem in atomic and molecular systems. calculation of wavefunction components in ursell-type expansion using quantum-field theoretical methods. *The Journal of Chemical Physics*, 45(11):4256–4266, 1966.

[35] I. Shavitt and R.J. Bartlett. *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory.* Cambridge Molecular Science. Cambridge University Press, 2009.

Bibliography

[1] D. R. Hartree. *Proc. Cambridge Phil. Soc.*, 24:89, 1928.

[2] V. Fock. Nherungsmethode zur lsung des quantenmechanischen mehrkrperproblems. *Zeitschrift fr Physik A Hadrons and Nuclei*, 61:126–148, 1930. 10.1007/BF01340294.

[3] C. Møller and M.S. Plesset. Note on an approximation treatment for many-electron systems. *Physical Review*, 46(7):618, 1934.

[4] J.W. Backus, R.J. Beeber, S. Best, R. Goldberg, L.M. Haibt, H.L. Herrick, R.A. Nelson, D. Sayre, P.B. Sheridan, H. Stern, et al. The fortran automatic coding system. In *Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability*, pages 188–198. ACM, 1957.

[5] B.W. Kernighan and D.M. Ritchie. *The C programming language.* Prentice Hall, 1988.

[6] Dupuis. M, Rys. J, and King H. F. Hondo. Quantum Chemistry Program Exchange, 11, 336338, 1977.

[7] W. J. Hehre, W. A. Lathan, R. Ditchfield, M. D. Newton, , and J. A. Pople. Gaussian 70. Quantum Chemistry Program Exchange, Program No. 237, 1970.

[8] J. Isaak. Standards-the history of posix: a study in the standards process. *Computer*, 23(7):89–92, 1990.

[9] B. Stroustrup. *The C++ programming language.* Addison-Wesley Longman Publishing Co., Inc., 1997.

[10] M. Dupuis and JD Watts. Parallel computation of molecular energy gradients on the loosely coupled array of processors (lcap). *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)*, 71(2):91–103, 1987.

[11] J.D. Watts and M. Dupuis. Parallel computation of the moller–plesset second-order contribution to the electronic correlation energy. *Journal of computational chemistry*, 9(2):158–170, 1988.

[12] G.D. Purvis III and R.J. Bartlett. A full coupled-cluster singles and doubles model: The inclusion of disconnected triples. *The Journal of Chemical Physics*, 76:1910, 1982.

[13] K. Raghavachari, G.W. Trucks, J.A. Pople, and M. Head-Gordon. A fifth-order perturbation comparison of electron correlation theories. *Chemical Physics Letters*, 157(6):479–483, 1989.

[14] M. S. Gordon and M. W. Schmidt. *Advances in electronic structure theory: GAMESS a decade later*, pages 1167–1189. Elsevier, Amsterdam, 2005.

[15] Cold war's end hits cray computer. *New York Times*, 1992.

[16] In supercomputers, bigger and faster means trouble. *New York Times*, 1994.

[17] DE Bernholdt, E. Apra, HA Früchtl, MF Guest, RJ Harrison, RA Kendall, RA Kutteh, X. Long, JB Nicholas, JA Nichols, et al. Parallel computational chemistry made easier: The development of nwchem. *International Journal of Quantum Chemistry*, 56(S29):475–483, 1995.

[18] A. Heinecke, M. Klemm, and H. Bungartz. From gpgpu to many-core: Nvidia fermi and intel many integrated core architecture. *Computing in Science & Engineering*, 14(2):78–83, 2012.

[19] Python. http://python.org/.

[20] Y. Shao, L.F. Molnar, Y. Jung, J. Kussmann, C. Ochsenfeld, S.T. Brown, A.T.B. Gilbert, L.V. Slipchenko, S.V. Levchenko, D.P. ONeill, et al. Advances in methods and algorithms in a modern quantum chemistry program package. *Phys. Chem. Chem. Phys.*, 8(27):3172–3191, 2006.

[21] J. M Turney, A. C Simmonett, R. M Parrish, E. G Hohenstein, F. Evangelista, J.T. Fermann, B.J. Mintz, L.A. Burns, J.J. Wilke, M. L Abrams, et al. Psi4: an open-source ab initio electronic structure program. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2011.

[22] Boost c++ libraries. http://www.boost.org/.

[23] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, et al. Lapack usersguide: Release 1.0. Technical report, Argonne National Lab., IL (United States), 1992.

[24] M. Born and R. Oppenheimer. Zur quantentheorie der molekeln. *Annalen der Physik*, 389(20):457–484, 1927.

[25] C.C.J. Roothaan. New developments in molecular orbital theory. *Reviews of modern physics*, 23(2):69, 1951.

[26] SF Boys. Electronic wave functions. i. a general method of calculation for the stationary states of any molecular system. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 200(1063):542–554, 1950.

[27] J.A. Pople and W.J. Hehre. Computation of electron repulsion integrals involving contracted gaussian basis functions. *Journal of Computational Physics*, 27(2):161–168, 1978.

[28] J. Rys, M. Dupuis, and H. F. King. Computation of electron repulsion integrals using the rys quadrature method. *Journal of Computational Chemistry*, 4(2):154–157, 1983.

[29] S. Obara and A. Saika. Efficient recursive computation of molecular integrals over cartesian gaussian functions. *The Journal of chemical physics*, 84:3963, 1986.

[30] M. Head-Gordon and J.A. Pople. A method for two-electron gaussian integral and integral derivative evaluation using recurrence relations. *The Journal of chemical physics*, 89:5777, 1988.

[31] G. Te Velde, F.M. Bickelhaupt, E.J. Baerends, C. Fonseca Guerra, S.J.A. van Gisbergen, J.G. Snijders, and T. Ziegler. Chemistry with adf. *Journal of Computational Chemistry*, 22(9):931–967, 2001.

[32] J.A. Pople. Molecular orbital methods in organic chemistry. *Accounts of Chemical Research*, 3(7):217–223, 1970.

[33] F. Coester and H. Kümmel. Short-range correlations in nuclear wave functions. *Nuclear Physics*, 17:477–485, 1960.

[34] J. Čížek. On the correlation problem in atomic and molecular systems. calculation of wavefunction components in ursell-type expansion using quantum-field theoretical methods. *The Journal of Chemical Physics*, 45(11):4256–4266, 1966.

[35] I. Shavitt and R.J. Bartlett. *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory*. Cambridge Molecular Science. Cambridge University Press, 2009.