Theor Chem Acc (2000) 103:349–352
DOI 10.1007/s002149900072

Theoretical
Chemistry Accounts

Perspective

Perspective on "Principles for a direct SCF approach to LCAO-MO ab initio calculations"

Almlöf J, Faegri K Jr, Korsell K (1982) J Comput Chem 3:385–399

Donald G. Truhlar

Department of Chemistry and Supercomputer Institute, University of Minnesota, Minneapolis, MN 55455-0431, USA

Received: 24 February 1999 / Accepted: 1 July 1999 / Published online: 4 October 1999

Abstract. The direct self-consistent-field (SCF) method recalculates all two-electron integrals each time they are needed in an SCF calculation. This perspective article discusses how the original paper on direct SCF by Almlöf et al. developed the principles by which this could be made efficient and thereby provided an example of the semantic approach to computational chemistry in which algorithm development and coding are not compartmentalized.

Key words: Computational chemistry – Direct self-consistent field method – Molecular orbital – Quantum mechanics – Two-electron integrals

Modern computational chemistry has come a long way. In the early days the advances often came by applying computers syntactically to algorithms that provided literal translations of theories developed without regard to computing platforms; however, such an approach would now be very old-fashioned. A modern computation has several distinct elements:

1. Fundamental equations. The underlying fundamental equations, sometimes called the first principles, are well known: Hamilton's equations, the Maxwell equations, the Liouville equation, and the Schrödinger equation were all well known by the 1920s and well integrated by the 1930s [1]. Feynman's path integral approach to quantum mechanics and statistical mechanics [2] was well appreciated by the 1960s.

2. Theory. Usually we do not solve the fundamental equations directly. We use a theory, for example, Hartree–Fock theory [3], Møller-Plesset perturbation theory [4], coupled-cluster theory [5], Kohn's [6, 7], Newton's [8], or Schlessinger's [9] variational principle for scattering amplitudes, the quasiclassical trajectory method [10], the trajectory surface hopping method [11], classical **S**-matrix theory [12], the close-coupling approximation [13–16], transition-state theory [17], variational-transition-state theory [18], self-consistent reaction-field theory [19], and so forth. Some of these theories date back to the 1930s, and new ones are continually being developed.

3. Algorithms. Most theories consist of differential or integral equations, sometimes integrodifferential equations, and these may be linear or nonlinear and are almost always multidimensional. Usually we cannot solve them by the classical, analytical methods of mathematics. Instead we reduce them to numerical algorithms, typically involving interpolation, extrapolation, quadrature, linear algebra, iteration, Monte Carlo sampling, and other general techniques. Prior to the days of automatic computing machines, this was the last step. For example, some of the early solutions to the Hartree–Fock equations were carried out using pencil and paper by Hartree's father, a retired sea captain, but he did not submit any articles about computational strategies to a scientific journal. Now, however, step 4 is a sine qua non, and the process of creating the program often entails as much research as developing the algorithm.

4. Software. To get numbers, one must convert the algorithm to a working computer program. In modern computational chemistry there are many opportunities for improved theories and improved algorithms, but more and more it is becoming clear that there is room for systematic, scientific progress in step 4 as well as in steps 2 and 3. In fact, treating step 4 as an afterthought no longer constitutes state-of-the-art work.

One of the first consequences of the increasing prominence of step 4 was to provide feedback to step 3 and promote the introduction of new algorithms. For example, the Monte Carlo method [20] of integration, unlike Gaussian quadrature or the Runge–Kutta algorithm, is hardly conceivable without computers. Other examples are the Cannon algorithm [21] or Strassen's algorithm [22, 23] for matrix multiplication or for the matrix multiplication steps in matrix inversion carried out by a matrix-times-vector formulation [22, 24]. In

comparing the Monte Carlo, Cannon, and Strassen algorithms to their traditional counterparts, one is inevitably forced to consider the details of a given computer architecture, especially its ability to perform computational steps in parallel or in pipelines and also to consider any communication bottlenecks that may arise in trying to take advantage of such parallelism. This has led to the emergence of parallel computation as a well-recognized field of study, with its own internal rules and dynamics. Parallel computation is a subfield of the broader emerging field of scientific computation.

At what point did the field of computational chemistry emerge as a distinct subfield of scientific computation? I would say that this occurred gradually, but certain milestones can be recognized. These milestones occurred whenever chemists recognized a new tradeoff (such as computation speed versus communication speed) that needed to be considered in applying state-of-the-art computer hardware to solve more challenging chemical problems. Usually (but not always) the challenge comes from increasing the size of the system.

When one takes a step back to gain a broader view, one finds that chemists, more than other computational scientists, have been stymied by memory bottlenecks. These were severe already in the 1960s, and by the 1980s it was very clear that many large-system calculations that were affordable in terms of computer time were not doable simply for lack of storage space to hold all the intermediate results. This was true especially for electronic structure calculations, where the aggravating quantities were two-electron integrals. As a student one quickly learned that the number of two-electron integrals in a calculation with N orbitals (this number scales as N^4 for large N). Then, if you knew how many integrals fitted on a 2400 foot tape, you could calculate how many tapes you needed. I have selected the paper by Almlöf, Faegri, and Korsell [25] for this perspective because it provided the first systematic attack on the memory bottleneck in computational chemistry and thereby, in my opinion, opened many people's eyes to the possibilities for progress if one rethinks the way one marshals the available computer resources to attack a computational problem. As one pursues this kind of thinking, the boundaries between theory, algorithms, and software actually begin to blur.

The desirability of blurring the boundary was made especially clear in the early days of vector computing by Kascic [26], who popularized the semantic approach to vectorization, which was the first form of parallel computing to benefit from systematic development. As Kascic pointed out, there are two ways to arrive at a vectorized algorithm. In the syntactic approach, we convert the physical problem into an algorithm (usually conceptualized as a scalar algorithm in those days), then we vectorize the algorithm. In the semantic approach, we proceed from the physical problem to the vectorized algorithm. It is like the difference between thinking in English and then translating into a foreign language or thinking directly in a foreign language. This merging of the design of step 3 (algorithm) and step 4 (computer code) is now well established for parallel computing. The paper by Almlöf et al. provides an analogous example of the benefits of redesigning the algorithm with the hardware capabilities in mind, but here the issue is the balance of computer speed with memory capacity, whereas in much parallel computation design the issue is balancing computer speed with communication speed. These issues are of course related since, if one is willing to tolerate slower access to data, one can store more of it. The capacity of the arithmetic registers is very low, level-2 cache is faster but is still very limited, high-speed memory is slower but larger, disk has a big latency cost but is even larger, and tape libraries still provide the largest, slowest storage capacity.

The method proposed by Almlöf et al. is called the direct self-consistent-field (SCF) method. It is the basis of virtually every large SCF-type calculation (Hartree–Fock or density functional theory) that is run today. It was motivated by the unsymmetrical rate of advance in central processor (CPU) technology and storage technologies. Originally CPU time was expensive, so each two-electron integral was saved and reused every time it was needed. Eventually though, as system size increased, these integrals had to be "out of core," typically on disks, and the time and effort to store them and the load that their retrieval placed on input/output capacity became the real bottleneck. The proposed solution is simply to recalculate the integrals every time they are needed rather than to store them and recycle them. The obvious drawback is that one increases the computer time, and the key advances were to ameliorate this problem. There were five elements in this:

1. As soon as one decides not to store and retrieve the integrals, one recognizes that one need not compute, store, retrieve, and use them in an order that makes retrieval efficient; rather one can restructure the algorithm for better CPU efficiency [27]. In particular one can switch to an integral-driven order of events. When an integral is calculated it should be used to the maximum possible extent.

2. In some implementations, direct methods place a greater premium on the use of symmetry [28] to identify integrals with permuted indices that are identical to each other and rearranging the algorithm to take advantage of that. (In other implementations the direct method diminishes rather than increases the importance of symmetry.)

3. Direct methods place a renewed emphasis on decreasing the number of SCF cycles by improving the iterative strategy [29].

4. Emphasis on the new integral bottleneck made one realize that not all integrals need be calculated after all [30]. One can devise efficient upper-bound estimates, and these can be used to prescreen the integrals. If the inexpensive estimate of the upper bound indicates that the integral will be negligible, the more expensive evaluation of the integral itself is omitted. Eventually a considerable amount of sophistication can be built into the prescreening process [31, 32].

5. Sometimes one does not need an integral, not because it is small, but because it will only be multiplied by small numbers in the rest of the calculation. In SCF calculations, the integrals are multiplied by density matrix elements; thus the magnitudes of the corresponding

density matrix elements are also used in deciding whether to calculate a particular integral. In particular one calculates a bound on the error in a Fock matrix element due to not calculating an integral [25]. Density-weighted integral estimates reduce the complexity of integral evaluation from $O(N^4)$ to $O(N^2)$ [33]. At an even higher level of sophistication one recognizes that only those integrals are required that are related to significant changes in the density matrix from one iteration to the next.

All these kinds of considerations get amplified as one proceeds to use direct methods in a wider context, for example, for energy gradients and Hessians [34], for linear scaling algorithms [35–38], or for relativistic effects [39]. Direct SCF methods are very well suited for parallel computation [40–42]. In addition the whole process of rearranging the algorithms makes one rethink the semantics of the problem in a very fruitful and stimulating way; such benefits would be missed if one simply syntactically translated the traditional algorithm into computer code.

Two other interesting algorithmic consequences of direct methods for quantum chemistry may be mentioned. First, viewing electron repulsion integrals as only intermediates on the way to the final results of interest (Fock matrix, energy, forces) has had snowballing consequences in electronic structure theory. The work of White and co-workers on fast multipole methods [43, 44] and **J**-matrix engines [45] are examples of more drastically redefining the intermediates. Second, optimizing the algorithm to the computer architecture can be usefully generalized as follows: given particular amounts of memory and disk, with given access rates, design the algorithm with the shortest time-to-solution. This has been pursued in so-called semidirect methods for second-order Møller–Plesset theory [46].

The ideas behind direct SCF calculations are also echoed in various ways in more far-afield areas of research. A very literal analog comes in basis-set approaches to scattering calculations [6–9] where we have explored [47] the tradeoff between recalculating integrals versus writing them to disk. A less obvious analog occurs in an optimized quadrature scheme [48] we developed. The widespread use of Gaussian quadrature formulas owes to their general efficiency for a wide variety of integrands. In some cases though there are tremendous cost savings by using more specialized quadrature formulas derived specifically for problem-specific functions that occur in a large number of integrals in a specific calculation. Modern computers can easily store large numbers of function-specific quadrature weights. If this allows one to decrease the number of quadrature points by a factor of 2 or 3 per dimension it can make not doable problems doable [49], for a cost of a few thousand storage locations. This would have been a considerable cost on the computer I used as a graduate student (total storage = 32K words), but is hardly worth a second thought now.

I hope this perspective conveys some of the benefits that can accrue from taking a more holistic view of the computational process and letting hardware considerations become more strongly coupled to algorithm design. This is the critical step in developing computational chemistry into a master discipline that integrates theoretical chemistry with scientific computation. I believe that the paper by Almlöf, Faegri, and Korsell provides a classic example of this kind of development.

Acknowledgements. I am grateful to Matt Challacombe, Martin Head-Gordon, Knut Faegri, Eric Schwegler, David Schwenke, and Peter Taylor for comments on the manuscript.

References

1. Tolman RC (1938) The principles of statistical mechanics. Oxford University Press, New York
2. Feynmann RP (1972) Statistical mechanics. Addison-Wesley, Reading
3. Fock V (1930) Z Phys 62: 795
4. Møller C, Plesset MS (1934) Phys Rev 46: 618
5. Bartlett RJ, Dykstra CE, Paldus J (1984) In: Dykstra C (ed) Advanced theories and computational approaches to the electronic structure of molecules, Reidel, Dordrecht, p 127
6. Kohn W (1948) Phys Rev 74: 1763
7. Truhlar DG, Abdallah J Jr, Smith RL (1974) Adv Chem Phys 25: 211
8. Newton RG (1966) Scattering theory of waves and particles, McGraw-Hill, New York
9. Schlessinger L (1968) Phys Rev 167: 1411
10. Karplus M, Porter RN, Sharma RD (1965) J Chem Phys 43: 3259
11. Tully JC (1998) In: Thompson DL (ed) Modern methods for multidimensional dynamics computations in chemistry. World Scientific, Singapore, p 34
12. Miller WH (1974) Adv Chem Phys 25: 69
13. Massey HSW, Smith RA (1933) Proc R Soc London Ser A142: 142
14. Wheeler JA (1937) Phys Rev 52: 1083
15. Lane NF, Geltman S (1967) Phys Rev 160: 53
16. Allison AC, Dalgarno A (1967) Proc Phys Soc 90: 609
17. Eyring H (1935) J Chem Phys 3: 107
18. Truhlar DG, Garrett BC (1984) Annu Rev Phys Chem 35: 159
19. Rinaldi D, Rivail JL (1973) Theor Chim Acta 32: 57
20. Siepmann JI (1999) Adv Chem Phys 105: 1
21. Cannon L (1969) Ph D thesis. Montana State University, Bozeman
22. Strassen V (1969) Numer Math 13: 354
23. Bailey D (1988) SIAM J Sci Stat Comput 9: 603
24. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) Numerical recipes in FORTRAN, 2nd edn. Cambridge University Press, Cambridge, p 95
25. Almlöf J, Faegri K Jr, Korsell K (1982) J Comput Chem 3: 385
26. Kascic MJ Jr (1983) Semantic and syntactic vectorization: whence cometh intelligence in supercomputing? Summer Computer Simulation Conference, Vancouver
27. Almlöf J, Faegri K Jr (1990) In: Carbo R, Klobukowski M (eds) Self-consistent field: theory and applications. Elsevier, Amsterdam, p 195
28. Almlöf JE (1997) Theor Chem Acc 97: 10
29. Pulay P (1980) Chem Phys Lett 73: 393
30. Almlöf J, Taylor PR (1984) In: Dykstra C (ed) Advanced theories and computational approaches to the electronic structure of molecules. Reidel, Dordrecht, p 107
31. Häser M, Ahlrichs R (1989) J Comput Chem 10: 104
32. Ruud K, Jonsson D, Norman P, Agren H, Save T, Jensen HJA, Dahle P, Helgaker T (1998) J Chem Phys 108: 7973
33. Strout DL, Scuseria GE (1995) J Chem Phys 102: 8448
34. Helgaker T, Jørgensen P (1992) In: Wilson S, Diercksen GHF (eds) Methods in computational molecular physics. Reidel, Dordrecht, p 353
35. Challacombe M, Schwegler E, Almlöf J (1996) J Chem Phys 104: 4685

36. Yang W, Perez-Jorda JM (1998) In: Schleyer PvR, Allinger, NL, Clark T, Gasteiger J, Kollman PA, Schaefer HF III, Schreiner PR (eds) Encyclopedia of computational chemistry. Wiley, Chichester, p 1496
37. White CA, Johnson BG, Gill PMW, Head-Gordon M (1994) J Chem Phys 230: 8
38. Ochenfeld C, White CA, Head-Gordon M (1998) J Chem Phys 109: 1663
39. Laerdahl JK, Save T, Faegri K Jr (1997) Theor Chem Acc 97: 177
40. Lüthi HP, Almlöf J (1993) Theor Chim Acta 84: 289
41. Lüthi HP, Mertz JE, Feynmann MW, Almlöf JE (1992) J Comput Chem 113: 160
42. Petterson LGM, Faxen T (1993) Theor Chim Acta 85: 345
43. White CA, Johnson BG, Gill PMW, Head-Gordon M (1994) Chem Phys Lett 230: 8
44. White CA, Johnson BG, Gill PMW, Head-Gordon M (1996) Chem Phys Lett 253: 268
45. White CA, Head-Gordon M (1996) J Chem Phys 104: 2620
46. Frisch MJ, Head-Gordon M, Pople JA (1990) Chem Phys Lett 166: 281
47. Schwenke DW, Mladenovic M, Zhao M, Truhlar DG, Sun Y, Kouri DJ (1989) In: Laganà A (ed) Supercomputer algorithms for reactivity, dynamics, and kinetics of small molecules. Kluwer, Dordrecht, p 191
48. Schwenke DW, Truhlar DG (1984) Comput Phys Commun 34: 57
49. Schwenke DW, Truhlar DG (1985) In: Numrich RW (ed) Supercomputer applications. Plenum, New York, p 215