Pseudospectral Hartree--Fock theory: Applications and algorithmic improvements

Murco N. Ringnalda, Mahfoud Belhadj, and Richard A. Friesner
Department of Chemistry, University of Texas at Austin, Austin, Texas 78712

(Received 13 March 1990; accepted 10 May 1990)

Several additions to the pseudospectral Hartree–Fock theory are described, including a localized least-squares procedure, various numerical cutoff algorithms, and calculation of all integrals in the diatomic frame. This pseudospectral method is tested on 23 molecules, ranging in size from two to twenty atoms (200 6-31G** basis functions). A direct comparison of accuracy and computational efficiency is made with the conventional electronic structure programs GAMESS, GRADSCF, GAUSSIAN 86, and GAUSSIAN 88. The pseudospectral code is shown to be up to nine times faster than any of the above programs for the molecules tested here; moreover, this timing advantage increases with molecular size, suggesting that *ab initio* calculations may soon be possible on large systems not accessible by the Roothaan–Hall procedure.

I. INTRODUCTION

The Roothaan–Hall (RH) method[1] for self-consistent field (SCF) calculations at the Hartree–Fock (HF) level of approximation has been a standard tool of computational quantum chemistry for some time. This technique has a number of outstanding features, not the least of which is the vast body of experience acquired by thousands of researchers over the past thirty years. Unfortunately, the computational effort is dominated by evaluation and usage of four-center, two-electron integrals (electron repulsion integrals) of the form

$$I_{\mu\nu\lambda\sigma} = \int\int \phi_\mu(\mathbf{r}_1)\phi_\nu(\mathbf{r}_1)\frac{1}{r_{12}}\phi_\lambda(\mathbf{r}_2)\phi_\sigma(\mathbf{r}_2)d\mathbf{r}_1\, d\mathbf{r}_2, \quad (1)$$

where the ϕ_i are basis functions, and r_1, r_2 represent the positions of two electrons. These integrals present five major problems in a standard RH code:

(i) The number of such integrals grows as N^4, where N is the total number of basis functions used.

(ii) The individual integrals are rather expensive to compute,[2] requiring a large number of floating point operations (flops).

(iii) The computation of such integrals is not easily adaptable to modern vector hardware.

(iv) All N^4 integrals are needed at each step of the SCF iteration procedure.

(v) Disk storage of $O(N^4)$ values is impossible on many machines even for medium-sized ($N = \sim 250$) calculations.

The last of these difficulties can be avoided by using the "direct" method of Almlöf, Faegri, and Korsell,[3] i.e., by recalculating the integrals at each iteration. This solution allows calculations on systems where the available disk space is less than N^4; note, however, that it magnifies the effect of (ii) and (iii). A recursive assembly of the Fock matrix F, usually used in combination with the direct method,[3-5] permits a reduction in the number of integrals used per iteration, since some integrals cease to contribute to the incremental improvements of F as convergence is reached.

Much effort has been devoted to minimizing the flops/integral ratio for the RH method.[6] Recently, Obara and Saika[7] have presented a recursive scheme for the calculation of two-electron integrals, which has been implemented by Head–Gordon and Pople[8] in the GAUSSIAN 88 system of programs. In addition to reducing the flop count, this method allows for some degree of vectorization of the otherwise purely scalar integral assembly; however, the vector loops involved are relatively short, and thus cannot take full advantage of the available hardware.

Thus progress has been made in recent years in regards to problems (ii)–(v). The advent of direct SCF methods has removed the computational load from the I/O subsystem and focused it upon the CPU, where modern computers excel. However, none of the improvements listed above directly address the most oppressive problem: that the number of integrals grows rapidly with basis set size. While cutoffs (the elimination of negligibly small integrals) can reduce the scaling of the method from N^4 for larger systems, RH calculations on large molecules remain difficult if not impossible. As an example, we point out a recently published coupled HF calculation[9] on the highly symmetric C_{60} ("Buckminsterfullerene") at the 6-31G* level (900 basis functions), which took 10 CPU days on a Cray X-MP/48. It should be clear that, if *ab initio* calculations on arbitrary, nonsymmetric large molecules are to be performed in an efficient manner, the scaling behavior of the RH method must be avoided.

In a series of recent papers,[10-15] we have reported the development of a new *ab initio* method which scales as N^3 (rather than N^4). This method is a synthesis of ordinary RH techniques (including such recent advances as direct SCF and Fock matrix updating) with the pseudospectral method,[16] originally developed for hydrodynamic simulations. For the basic equations of this pseudospectral Hartree–Fock theory we refer the reader to Refs. 10–12; we point out here only that the pseudospectral approach involves the use of both a spatial grid *and* ordinary quantum chemical basis sets, thereby eliminating two-electron integrals while retaining the accuracy of the RH (spectral) method.

In addition to presenting the basic theory, Refs. 10 and 11 included results from pseudospectral HF calculations on atoms and diatomic molecules, using Slater basis functions. Reference 12 applied the method to the water molecule, using 6-31G** Gaussian basis sets.[17] In Ref. 13, an algorithm for automatic construction of molecular grids was suggested. Such an algorithm is required for calculation on arbitrary polyatomics; that provided in Ref. 13, although amended slightly in the present paper, has proven remarkably robust.

We introduced, in Ref. 14, an inexpensive Newton–Raphson convergence scheme, a pseudospectral direct SCF procedure, multigrid techniques, and other improvements to the original algorithm. The method was tested on the glycine molecule (10 atoms, 100 basis functions) at the 6-31G** level, and a direct comparison with RH programs showed substantial reductions in CPU times, while maintaining adequate accuracy.

Most recently,[15] we extended the method beyond the HF approximation, including electron correlation effects through the generalized valence bond (GVB) formalism.[18] Reference 15 also includes the first tests of the method with other Gaussian basis sets, with effective core potentials, and on a second row atom (Si).

In this paper, we present several additional improvements to the pseudospectral algorithm and compare the current pseudospectral code to other available electronic structure codes in terms of accuracy and computational efficiency.

In Sec. II, changes to the grid generation algorithm are detailed, and a grid sorting procedure is presented. These changes reduce the directional dependence of the atomic grids, allow faster assembly of the $\mathbf{R}^\dagger\mathbf{wR}$ matrix used in the least-squares procedure, and facilitate the use of various cutoffs throughout the program.

Section III lists major changes to the dealiasing/least-squares algorithm which yield greater accuracy, reduced CPU time, and reduced memory usage, at the cost of more complex coding.

The three-center, one-electron integrals are now calculated and used in the diatomic frame. This reduces the number of flops per integral, but requires transformation of several matrices into or out of the diatomic frame. These changes are presented in greater detail in Sec. IV below.

Finally, in Sec. V, we present results of HF calculations on 23 molecules, ranging in size from H_2 to glutamine ($C_5N_2O_3H_{10}$). We compare the pseudospectral results to those of GAMESS, GRADSCF, GAUSSIAN 86, and GAUSSIAN 88, and note that not only is the pseudospectral code substantially faster, but that the scaling of CPU time with basis set size is considerably better.

II. GRID GENERATION AND ITS ALGORITHMIC IMPLICATIONS

In this section we detail changes and additions to the algorithm for automatic atomic grid generation of Ref. 13, as well as our use of the resultant grids.

We have added, as an option, 29 angular gridpoint distributions by Lebedev[19–21] and Stroud[22] to the existing angular grids which were described in Ref. 13 and are based upon the work of Glatzmaier.[23] The Glatzmaier grids consist of points arranged in latitudinal rings on the sphere, and the quality of the numerical quadrature was found to vary depending upon the orientation of the polar axis. The newer grids are expected to alleviate this problem, as they have a much more even distribution of points about the sphere. The grids discovered by Lebedev have octahedral symmetry and range in size from 38 to 302 points, while those given by Stroud are generally smaller and have either octahedral or icosahedral symmetry.

In our previous paper,[14] we used four sizes of molecular grids in the SCF convergence scheme: a very coarse grid, used only to construct the approximate Hessian for the Newton–Raphson procedure; a medium sized grid used for most of the SCF iterations; an "ultra-fine" grid used for a single iteration, after which Fock matrix updating was used; and the fine grid, intermediate in size between the medium and ultrafine grids, which was used only on the first updating step. We have changed this scheme only in that we have eliminated the fine grid. Due to changes in the least-squares procedure described below, as well as further optimization of grid parameters, the medium grid is now of sufficient quality to take the place of the fine grid. This results in a considerable savings of CPU time, since now neither the $\mathbf{Q}$ matrix nor the integrals for the fine grid need to be computed.

In Ref. 14, we presented an algorithm for constructing the $\mathbf{R}^\dagger\mathbf{wR}$ matrix in an efficient manner, based upon condensation of basis and dealiasing sets into blocks of *s*, *sp*, *spd*, and *spdf* functions (K, L, M, and N shells). We outline here a modification of this scheme which also takes advantage of the special structure of the atomic grids. In particular, the exponential part of a Gaussian function depends only upon the radius $\mathbf{r}$ from its center, and not upon any angular coordinates. Since the atomic grids are arranged in radial shells, the exponential function needs only be evaluated N_r times (where N_r is the number of radial shells), rather than once for each gridpoint. This is of course only true when the centers of the Gaussian and the atomic grid coincide.

Given a set of Gaussians centered on atom A, another set on atom B, and a (truncated) atomic grid centered on atom C, construction of the relevant piece of the $\mathbf{R}^\dagger\mathbf{wR}$ matrix proceeds by one of three paths:

(i) If C is different from both A and B, then the method of Ref. 14 is used, i.e., we construct all possible products of the radial functions and all of the angular functions necessary at each gridpoint, then sum over the grid. If the average number of angular functions per exponential pair is N_a, each radial shell contains an average of N_s gridpoints, and there are N_A and N_B unique exponents on atoms A and B, respectively, then the cost of this calculation is $N_a N_r N_s N_\mathrm{A} N_\mathrm{B}$. The fraction of atom triplets A, B, C which must be done by this method is approximately $(N_\mathrm{atom} - 1)/(N_\mathrm{atom} + 1)$.

(ii) If A, B, and C are identical, we need only calculate the exponential product at each of the N_r shells, then sum the angular and radial parts over shells. The rate-determining step here costs $(N_a N_r N_A^2)/2$, and thus a factor of N_s is saved.

(iii) If A and C coincide, but A and B do not, the calculation is carried out in two steps. First, we sum the angular functions with the radial functions from atom B, over the gridpoints in each radial shell. The resultant matrix is multiplied by the matrix of exponentials from atom A, evaluated at each radial shell. The cost here is $N_a N_r N_s N_B + N_a N_r N_A N_B$.

For the finest grid used, N_A and N_B average ~ 10, N_r is 18, and N_s is ~ 50. This suggests a reduction in CPU time from case (i) of a factor of 50 for case (ii) and a factor of about 10 for case (iii), and these reductions were in fact achieved. Note that cutoffs based upon the exponential decay of basis functions eliminate a large percentage of case (i) calculations, thus increasing the importance of the above improvements for cases (ii) and (iii) for which cutoffs are not so effective.

To facilitate the implementation of the various cutoff schemes described in this paper, we found it necessary to sort the atomic grids into sets of spatially contiguous blocks. The algorithm for this reordering of the radially generated grid is as follows. Suppose the target size of each contiguous block is N_{buck} gridpoints. The first block is made up of the innermost radial shells, as many as will fit within the N_{buck} limit. For the next radial shell(s), a relatively small set of evenly spaced points ("nuclei") is generated, and each gridpoint in the radial region is assigned to the "nucleus" to which it is closest. We currently use as nuclei the vertices of a dodecahedron. A list of neighboring nuclei is maintained, and neighbors are combined if the number of gridpoints in the combined block does not exceed N_{buck}. The process is repeated for the shells at larger radii. More than one radial shell may be processed at once, depending upon the number of gridpoints in each shell.

Having sorted the grid, we then calculate the minimum distance from each gridblock to each atomic center (by simply examining each gridpoint in turn), and store these values for use in the cutoff schemes described in Secs. III and IV below. The minimum distance computation, together with the sort, takes less than 0.1 CPU seconds/atom and is therefore computationally trivial.

III. LOCALIZED LEAST-SQUARES PROCEDURE

In order to transform the pseudospectral Fock operator back to spectral (atomic orbital) space, the $N \times M$ matrix (M is the number of gridpoints)

$$\mathbf{Q} = \mathbf{S}[\mathbf{R}^\dagger \mathbf{w} \mathbf{R}]^{-1} \mathbf{R}^\dagger \mathbf{w} \tag{2}$$

must be formed [$\mathbf{S}$ in Eq. (2) is the analytic overlap matrix, while $\mathbf{R}$ is the matrix of basis and dealiasing functions evaluated at the gridpoints, and $\mathbf{w}$ contains the grid weights]. As originally formulated,[12] construction of the least-squares matrix $\mathbf{Q}$ presented two computational problems. The first was the effort required to assemble the matrix

$$\mathbf{C} = \mathbf{R}^\dagger \mathbf{w} \mathbf{R} \tag{3}$$

which would scale as N^3, with a very large prefactor, if done directly. This problem has been addressed in Ref. 14 and in Sec. II of this paper; the time required for the assembly of C now grows more slowly than N^2.

The second difficulty is the size of C. For N basis functions, roughly $6N$ dealiasing functions are required, so that C has $\sim 0.5*(7N)^2$ unique elements. Although the entire matrix could be stored in core memory for even the largest of our current test cases ($N = 200$), this would not be true for larger N.

One possible remedy would be to simply store the matrix on disk and use an out-of-core routine for the inversion. The inversion itself, however, requires $O(N^3)$ operations, and if the size of C is allowed to grow without bound, this step will eventually dominate the computation time.

Instead, we choose to localize the least-squares procedure, forming and inverting several (smaller) C matrices, each of which is used to form one or more rows of Q. Two separate algorithms are employed here, one for short-range basis functions (defined at present as those having a radial exponent larger than 0.5) and another for the more extended functions. We first describe the short-range scheme; the other follows in Sec. III B.

A. Assembly of Q, short range

For the 6-31G** basis set, nearly all functions qualify as short range; only the $2s$ of hydrogen and the $3sp$ of first row atoms have exponents smaller than 0.5. Each atomic set of short-range functions is treated separately, i.e., N_{atom} least-squares solutions are obtained, each projected back onto the basis set using only the relevant rows of the overlap matrix. The advantage in this method is that, as might be expected, functions which do not overlap the target functions (the atomic block of short-range functions) need not be included in the fitting matrix. In fact, we find that functions having nonzero overlap with the target functions, but whose centers lie outside of the range can also be excluded. This reduces the size of the C matrix considerably.

We further enforce the exclusion of functions on distant centers by multiplying the diagonal matrix of grid weights w by $\exp(-\lambda r^2)$, where r is the distance from the gridpoint to the common center of the target functions and λ is the smallest of the radial exponents of those functions. This has the effect of forcing a better fit near the nucleus, where the basis functions have larger amplitude. Additionally, it allows all gridpoints lying beyond some cutoff radius to be ignored. The minimum distance data described in Sec. II are used to determine which particular gridblocks are to be kept.

To assemble the fitting matrices in an automatic way, we maintain a list of "neighbors" for each atomic center. For example, all centers closer than 3.0 bohr are defined as nearest neighbors, those at a distance of 3.0–5.0 are next-nearest neighbors, etc. Dealiasing sets are then optimized for both atoms and relative positions, e.g., all carbon atoms appearing as next-nearest neighbors of the central atom (the center of the particular localized least-squares calculation) have the same dealiasing set, though this set might be different than the carbon nearest neighbor set. Tests show that a minimal set of functions can be used for next-nearest neighbors, and that next-next-nearest neighbors can be ignored.

The net result is to create N_{atom} short-range C matrices, each constructed from only 100–400 functions and no more than 14 000 gridpoints. Because these matrices have a fixed size, the computational effort here grows linearly with the number of atoms. Furthermore, this procedure increases the accuracy of the least-squares fit, since the fitting matrices constructed in this way can be optimized individually without causing instabilities in the overall procedure.

B. Assembly of Q, long range

Having removed the short-range functions in the manner described above, the simplest procedure to deal with the extended functions is to simply regroup them into a single block, and assemble one **C** for all. The number of fitting functions necessary is somewhat smaller due to the absence of the short-range functions, but only by 20%–30%, and so the size of this long-range **C** matrix eventually becomes unreasonable.

One could adopt the same strategy that was used successfully for the more localized functions, i.e., create individual **C** matrices for each atom and use an exponential weighting factor. This would again result in linear scaling with molecular size, but with a very large prefactor. The difficulty here is that the radial functions die off very slowly, so that functions from dozens of neighboring atoms might need to be included in each fitting matrix.

In this case the multiplication of grid weights by the exponential envelope does little good: the envelope function itself is rather flat. Abandoning it brings two rewards: (i) a separate fitting procedure is no longer necessary for *every* atom, and (ii) elements of **C** which occur in more than one localized procedure can be reused (i.e., the matrix **w** is the same for all **C**). The latter is quite important from the standpoint of computational efficiency, since any division of the long-range functions will create **C** matrices which have a large number of functions in common. The former is useful because, e.g., a hydrogen atom is likely to have very nearly the same list of neighbors as does the atom to which it is bonded. Given that fitting functions are assigned based upon the neighbor list, it makes sense to combine the **C** matrix for the hydrogen with that of its neighbor, as the two matrices are likely to be nearly identical.

The algorithm we use, then, is as follows. The starting point is N_{atom} atomic long-range blocks (i.e., N_{atom} potential **C** matrices). Each block is assigned a set of fitting functions as in the short-range algorithm (the dealiasing sets used here are different, of course: they include smaller exponents and functions on more distant centers). The merger of adjacent blocks is then considered. The set of fitting functions for a combined block is the union of the two original sets; this must be sufficiently small that the resultant **C** matrix will fit into the available memory. If this condition is not satisfied, the merger is rejected. All possible combinations are examined before each merger so that the pair of blocks with the most common elements can be joined. This process is repeated until all possible mergers are rejected by size limitations. For small molecules, this results in a single long-range block. Should there be more than one long-range block at this point, the order of calculation of the **C** matrices is rearranged so as to minimize the number of elements which must be saved after each matrix assembly.

We currently restrict the **C** matrices to have dimension 880 or less. This is considerably larger than the short-range blocks, but is still manageable. Note also that matrix inversions of this size are relatively inexpensive and do not dominate the CPU time for this step. Most importantly, the maximum dimension of **C** given above does not increase with molecular size.

C. Symmetrization of the Fock matrix

An issue related to the foregoing is the symmetrization of the Fock matrix. The pseudospectral Fock matrix is not necessarily Hermitian, and we previously symmetrized it by averaging the off-diagonal elements.[14] Comparison of pseudospectral and spectral Fock elements shows that accuracy can be gained in some cases by preferentially choosing either F_{ij} or F_{ji} and discarding the other. In particular, the short-range rows of **Q** are now more accurate (because of the exponential weighting and individual least-squares solutions) than the long-range rows. Furthermore, among the short-range rows, those with the largest exponents are the most accurate. To take advantage of this, we establish a priority schedule for the basis functions, assigning an integer weight to each function. If the priority number of function i is higher than that of j, F_{ij} is kept and F_{ji} discarded. For functions of equal priority, F_{ij} and F_{ji} are averaged as before. For the 6-31G** basis set, we assign the the highest priority (4) to the heavy atom $1s$ functions, and the lowest (0) to the long-range functions. The first row $2sp$ block is given priority 1, the $3d$ priority 2, and the hydrogen short-range functions have priority 3. This scheme has the additional advantage of saving some time in the final assembly of the Fock matrix, since we no longer need to compute all N^2 elements.

IV. THREE-CENTER, ONE-ELECTRON INTEGRALS

Our original integral package was a fairly straightforward implementation of the formulas given by Taketa, Huzinaga, and O-Ohata.[24] All integrals were calculated at once in preprocessing and stored on disk. This procedure was not grotesquely inefficient because vectorization was possible over the gridpoints, and because the pseudospectral method does not require four-center, two-electron integrals. We have completely rewritten the integral code, implementing three major new features: (i) vectorization of calculations which do not include a grid index; (ii) recalculation of all integrals by default, with optional storage of some or all; and (iii) computation of all integrals in the diatomic frame.

Of these, the most important advance is the third. A huge reduction in the number of operations necessary per integral is obtained when the problem is transformed from the molecular coordinate system to the two-dimensional (cylindrical) space which is natural to the three-center integrals (recall that the pseudospectral integrals are indexed by two basis functions and a gridpoint). There is of course a price to be paid: the results must be transformed back to the molecular frame. Instead of transforming the integrals themselves (an operation which would scale like N^2M, where M is the number of gridpoints), we choose to transform the eigenvectors, the grid coordinates, and the $M \times N$ matrix **RP** (the product of the matrix of basis functions evaluated on the grid **R** and the density matrix), assemble the Coulomb and exchange operators in the diatomic frame, and back-transform the resultant $N \times M$ exchange matrix to the molecular frame (note that the pseudospectral Coulomb operator is simply a vector on the grid, and is thus invariant). The CPU time required for these transformations is not negligible; they in fact consume 9% of the total time for glutamine. However, this is more than offset by the savings in the calculation of the integrals themselves, especially for "direct" SCF.

The structure of the Coulomb and exchange operator assembly, including integral calculation, is as follows. The integral package is called, for a given gridblock, once for each pair of atoms. The integral driver loops over blocks of basis functions (e.g., 1*s*, 2*sp*, 3*d*, etc.) on each atom and calls a specialized routine for each type of function block pair. Within the latter routine, the functions[25]

$$F_v(t) = \int_0^1 u^{2v} e^{-tu^2} du \tag{4}$$

are constructed, and the various quantities which do not depend upon the grid ("constants") are assembled, vectorizing over the list of pairs of primitive Gaussians which have survived cutoffs. The F_v functions and the "constants" are combined by matrix multiplication, and the final assembly of the potential integrals is vectorized over the grid. A routine to compute the pseudospectral exchange operator from the integrals is then called. Finally, the Coulomb operator is calculated, and control is returned to the main program.

One use of the minimum distance data described in Sec. II is in the computation of the primitive functions $F_v(\gamma r^2)$ on the grid. Here γ is the exponent of a product Gaussian, and r is the distance from the product center to the gridpoint. Our general approach to this calculation is given in Ref. 12; we modify it here to take advantage of the fact that $F_v(t) \propto \mathrm{erf}[sqrt(t)]$, and that $\mathrm{erf}(x)$ goes rapidly to 1.0 as x increases. We set a cutoff of 16.5 for $\gamma r_{\min}^2$ $[1-\mathrm{erf}(sqrt(16.5)) < 10^{-8}]$; for the gridblocks which meet this condition, the table look up and interpolation is avoided, and the F_v reduce to the asymptotic formula

$$F_v(t) = \frac{(2v-1)!!\pi^{1/2}}{2^{v+1} t^{v+1/2}}. \tag{5}$$

This cutoff criterion is satisfied 69% of the time for glycine and 77% for glutamine.

In addition to the asymptotic form for the functions F_v, cutoffs in the integral package include a simple check of the size of the exponential prefactor, and estimates of the eventual contribution to the Fock matrix, based upon the prefactor, the density matrix, and the intermediate matrix **RP** defined above.

We have added an option which allows storage of the most expensive integrals while recalculating the rest. The "semidirect" method, pioneered by Häser and Ahlrichs,[5] allows optimization of the CPU/disk storage tradeoff. In our implementation, we avoid the need for estimates of CPU time for each type of integral by the simple expedient of calling timing routines before and after the first calculation of any integral block. If the time returned is greater than a (machine-dependent) cutoff, the set of integrals is stored, and a flag is set. The fact that the timing routines and the cutoff are machine dependent is not a drawback to this method, since the problem being solved, optimization of hardware resource usage, is inherently machine dependent. Figure 1 shows that this scheme works, e.g., storage of 20% of the integrals saves 50% of the extra CPU time. This entire issue is somewhat less important for the pseudospectral method than for purely spectral codes, since the difference in CPU time between fully standard SCF and fully direct SCF is only $\sim 15\%$ (Häser and Ahlrichs report an increase of 30% in CPU time for an 88% direct case).

V. RESULTS

In this section, we present a direct comparison of the performance of our pseudospectral HF code with that of the conventional programs GAMESS, GAUSSIAN 86, GAUSSIAN 88, and GRADSCF. Considering the complexity of electronic structure codes (some include over 100 000 lines of FORTRAN), there are a surprising number of different RH programs in existence. We chose the four listed above because they are widely known, state of the art, and readily available. All results reported below are from single-geometry HF calculations using the standard 6-31G** basis sets.

We tested some 23 molecules for accuracy against GAUSSIAN 86, and ran all five programs on the largest six of these in order to compare computational efficiency. Our program at present does not take advantage of molecular symmetry in any way, though we intend to rectify this in the near future. For this reason, symmetry was explicitly turned off in the four spectral programs. The largest of the molecules tested here are, in any case, non-symmetric.

Unfortunately, we did not have access to all four RH programs running on a single computer. GAUSSIAN 86 and GRADSCF were tested on a Cray X-MP/24, while GAMESS and GAUSSIAN 88 were run on a Cray X-MP EA/14se. Tests on our own program indicate that CPU times vary by no more than 4% between the two machines.

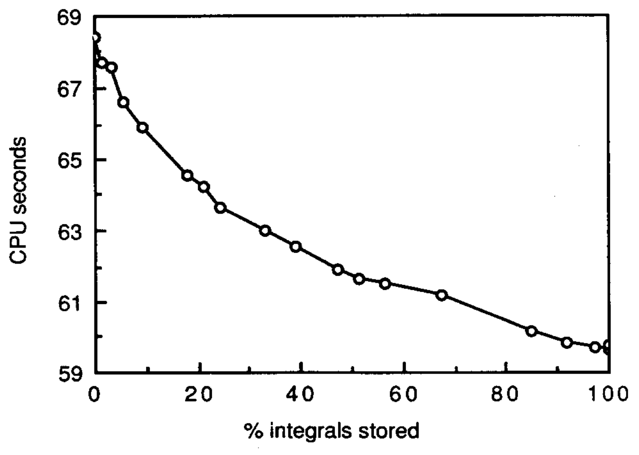

FIG. 1. CPU s vs percentage of integrals stored on disk, for pseudospectral HF calculations on glycine, on a Cray X-MP/24.

TABLE I. Comparison of pseudospectral and conventional programs: accuracy.[a]

Molecule	Reference[b]	E_{RH}[c] (hartrees)	Pseudospectral error (hartrees)	Pseudospectral error (kcal/mol)
H_2	26	− 1.131 334	− 0.000 004	− 0.002
N_2	26	− 108.943 949	− 0.000 002	− 0.001
O_2	26	− 149.529 475	0.000 128	0.080
HF	26	− 100.011 351	0.000 044	0.028
CO	26	− 112.737 877	0.000 077	0.048
H_2CO	26	− 113.869 742	0.000 088	0.055
H_2O	26	− 76.023 615	− 0.000 007	− 0.004
H_2O_2	26	− 150.770 784	− 0.000 028	− 0.018
HCN	26	− 92.865 968	0.000 056	0.035
HOCN	26	− 167.708 663	0.000 103	0.065
CH_3F	26	− 139.038 782	− 0.000 092	− 0.058
C_2H_2	26	− 76.821 837	− 0.000 049	− 0.031
C_2H_3OH	26	− 152.892 173	0.000 026	0.016
CH_3OH	26	− 115.045 721	− 0.000 045	− 0.029
CH_2NH	26	− 94.035 706	0.000 074	0.046
NH_3	26	− 56.195 375	− 0.000 009	− 0.006
NH_2F	26	− 154.959 174	0.000 002	0.001
Cyclobutadiene		− 153.634 912	0.000 105	0.066
Glycine 0°	27	− 282.844 468	− 0.000 093	− 0.058
Glycine 150°	27	− 282.841 397	0.000 156	0.098
Glycine 180°	27	− 282.841 671	0.000 080	0.050
Benzene		− 230.701 687	− 0.000 001	− 0.001
Uracil	28	− 412.479 487	0.000 020	0.013
Glycylglycine	29	− 489.550 218	0.000 022	0.014
Glutamine	30	− 528.646 752	− 0.000 025	− 0.016

[a] All data are from 6-31G** single-geometry HF calculations w/o symmetry on a Cray X-MP/24.
[b] Source of molecular geometry.
[c] Roothaan–Hall total energies from GAUSSIAN 86.

In addition to the Crays, we have successfully run our code on minisupercomputers made by Alliant, FPS, and Stellar. In this paper we present results only for the Crays, on which a direct comparison of the various programs was possible. However, reductions in CPU usage appear to be of similar magnitude on the smaller machines. Furthermore, disk storage on these machines is relatively limited; the RH programs which store integrals on disk could not have been tested for the largest molecules.

A. Accuracy

In Table I we report total energies for 23 molecules, as determined by the pseudospectral program and by GAUSSIAN 86. The molecules tested range in size from two to twenty atoms (10 to 200 6-31G** basis functions). The geometries for most of the smaller molecules were obtained from Ref. 26, while the nuclear coordinates for the largest were taken directly from neutron diffraction data. The three glycine geometries are those we previously tested in Ref. 14. The source of geometry information for each molecule is listed in Table I, with two exceptions. Benzene and cyclobutadiene were constructed from arbitrary bond lengths and angles, in order to verify the efficacy of the pseudospectral program at non-equilibrium geometries.

The molecules in Table I contain only hydrogen and first row atoms. We have not yet optimized atomic grids and dealiasing sets for atoms further down the Periodic Table,

TABLE II. Comparison of pseudospectral and conventional programs: CPU timings (in s).[a]

Molecule (basis functions)	GAMESS[b]	GAUSSIAN 86	GAUSSIAN 88[b,c]	GRADSCF	Pseudospectral[c]
Cyclobutadiene (80)	98	152	209	88	44
Glycine (100)	267	416	503	194	68
Benzene (120)	439	702	634	367	111
Uracil (140)	760	1297	1242	611	143
Glycylglycine (175)	1990	3729	2067	1302	245
Glutamine (200)	3367	6667	3458	(d)	378

[a] All data are from 6-31G** single-geometry HF calculations w/o symmetry.
[b] GAUSSIAN 88 and GAMESS were tested on a Cray X-MP EA/14se, while all other timings are from a Cray X-MP/24. Timings for pseudospectral code differ by no more than 3 s between the two machines.
[c] GAUSSIAN 88 and the pseudospectral program were run in 'direct' mode; GAMESS, GAUSSIAN 86, and GRADSCF stored integrals on disk.
[d] Calculation not possible due to GRADSCF sort limit of 200 million elements.

though that work is now in progress. While a fairly limited number of atoms appear in these molecules, a wide variety of molecular structures are represented. Included are single, double, and triple bonds, four- and six-membered rings, homo- and heteronuclear rings, nonsymmetric and highly symmetric molecules, planar and globular systems, zwitterions (glycylglycine and glutamine), and a number of different organic functional groups.

For each of the 23 molecules, the difference between the pseudospectral and RH results is less than 0.1 kcal/mol (0.000 16 hartrees), and for most it is considerably smaller. We believe that this is adequate accuracy for most if not all applications in which the RH method is currently used.

An important point is that the grids and dealiasing sets for each molecule were chosen in a completely automatic way, i.e., we now have, in addition to the 6-31G** basis set for (e.g.) carbon, three 6-31G** carbon grids (coarse, medium, and fine) and similarly, three 6-31G** carbon dealiasing sets. Of course, our grids and dealiasing functions are expected to change in the future; optimization is far from complete.

Because the grids and dealiasing sets are currently nonoptimal, we expect the accuracy of the pseudospectral program to increase in the future. It is interesting to compare the current results with those of Refs. 10 and 12 and to note that, as the molecular size increased, the pseudospectral accuracy increased also. This is not, of course, a feature of the method, but rather an indication of our growing understanding of the complicated interrelationships between basis sets, grids, and dealiasing functions.

Finally, we note that the accuracy of the pseudospectral method is tied to the size of the finest grid. Increasing the number of gridpoints (in a controlled fashion) will increase the accuracy (i.e., the agreement with the RH result), at some cost in CPU time; conversely, if the current level of accuracy is unnecessarily high, the grid size can be reduced, with a resultant reduction of cost. We intend eventually to develop a variety of grids, so that the desired level of agreement with RH may be input by the user along with the molecular geometry and choice of basis sets.

B. CPU times

In Table II we present CPU times for HF calculations on a variety of medium-sized molecules. As noted above, GRADSCF and GAUSSIAN 86 were run on a different machine than the other conventional programs. The results for our program in Table II are from the X-MP/24; the timings on the X-MP EA/14se differed from these by no more than 3 CPU s in any case. The pseudospectral program can be seen to achieve substantial advantages for every molecule listed. More importantly, the advantage increases with molecular size, from a factor of 2 (vs GRADSCF) for cyclobutadiene to nearly a factor of 9 (vs GAMESS) for the largest molecule, glutamine.

The data in Table II give some indication of the superior CPU scaling of the pseudospectral method with basis set size. In order to examine this more closely, in Table III we divide the pseudospectral CPU times from Table II into three parts: (a) preprocessing time, which is dominated by the assembly of the least-squares matrix **Q**, but also includes such tasks as basis set input and indexing, overlap and one-electron matrix computation, and grid generation; (b) integral calculation and assembly of Coulomb and exchange operators; and (c) Fock matrix preprocessing and assembly.

The last of these is dominated (for glutamine) by four operations: assembly of the intermediate matrix **RP**, the transforms to and from the diatomic frame, and the multiplication of the pseudospectral Fock operator by **Q**.

The CPU times in Table III are shown graphically in Figs. 2(a)–2(c), along with lines indicating the corresponding times for hypothetical programs that scale as N, N^2, or N^3. It should be noted that because the molecules listed in Table II took varying numbers (8–10) of Newton–Raphson iterations to converge, the actual CPU times were scaled in Table III to a constant eight iterations. Thus the sums of CPU times from Table III may not equal the actual total reported in Table II.

It can be seen from Fig. 2(a) that the preprocessing part of the program scales somewhat better than N^2. This suggests both that the algorithmic changes of this work and of Ref. 14 were successful, and that this part of the calculation will consume a negligible part of the CPU time for larger molecules. We expect, based upon the discussion in Sec. III, that this part of the calculation will scale linearly with molecular size in the asymptotic limit. Clearly, that limit is beyond 20 atoms, and Fig. 2(a) shows an inconsistent scaling behavior. This can be attributed to two factors:

1. Benzene and uracil are planar systems, while glycylglycine and (especially) glutamine are more globular. A globular molecule requires larger fitting matrices, since each atom has a larger number of neighbors. Thus a more substantial increase in CPU time should be expected from uracil to glycylglycine than from benzene to uracil.

2. The decomposition of the long-range **C** assembly begins with glycylglycine. Uracil requires only one long-range least-squares matrix, glycylglycine two, and glutamine three. Since the number of these matrices is limited to N_{atom}, this must eventually scale linearly, even though the growth in this region is faster than that.

The time used by the portion of the program that calcu-

TABLE III. Pseudospectral CPU timings by task (in s).[a]

Molecule (basis functions)	Preprocessing[b]	Integrals[c]	Fock assembly[d]
Cyclobutadiene (80)	20.1	17.2	7.0
Glycine (100)	28.7	26.7	12.5
Benzene (120)	43.6	46.2	20.3
Uracil (140)	48.4	55.4	28.3
Glycylglycine (175)	82.3	94.0	58.5
Glutamine (200)	121.1	141.6	88.7

[a] All data are from 6-31G** single-geometry HF calculations w/o symmetry, on a Cray X-MP/24. Times are scaled to a constant eight Newton-Raphson iterations (see text).
[b] Includes preprocessing steps and assembly of Q matrix.
[c] Includes integral calculation and assembly of Coulomb and exchange operators.
[d] Includes Fock matrix assembly and diatomic frame transformations.

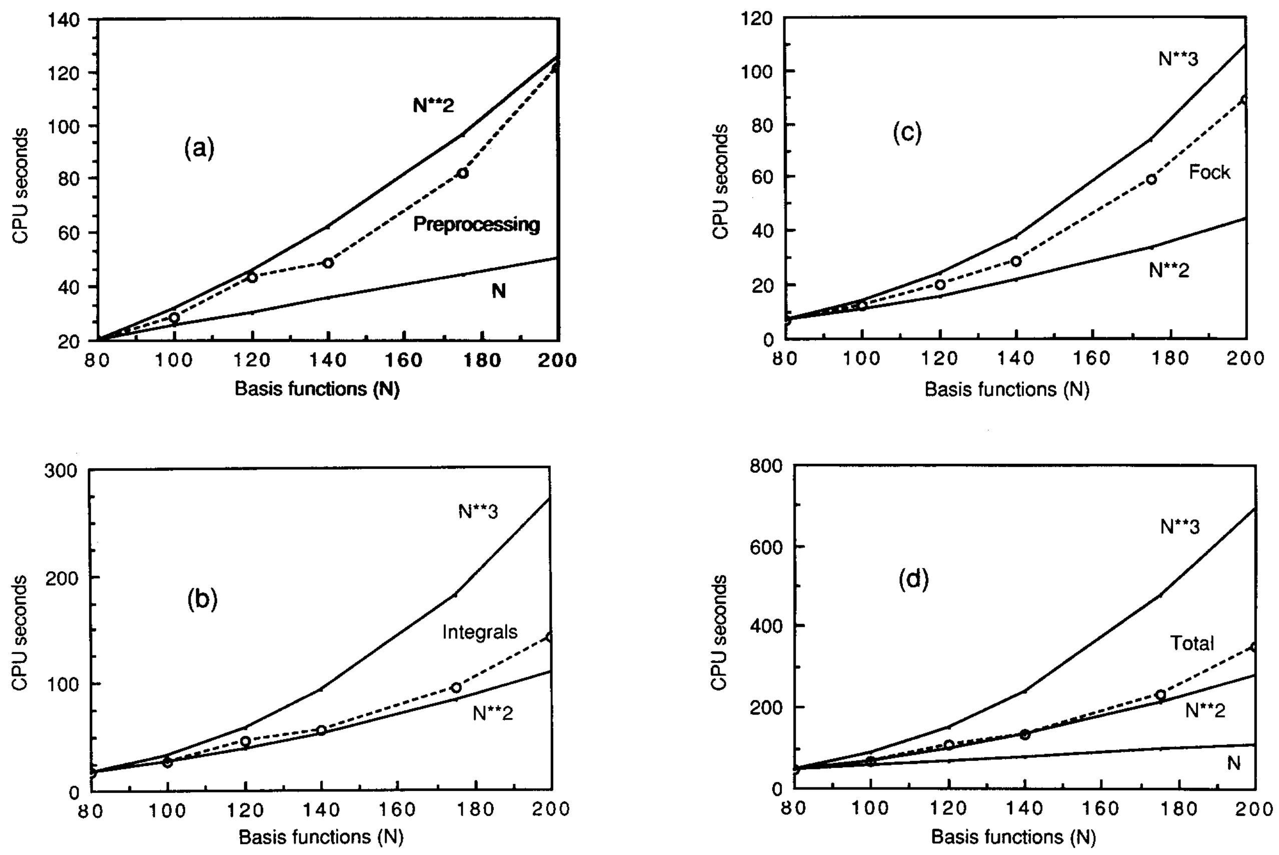

FIG. 2. Scaling of pseudospectral CPU time with basis set size, by task. (a) Preprocessing; (b) integral calculation and use; (c) Fock matrix assembly; and (d) total [sum of a, b, and c]. Solid lines represent linear, quadratic, and cubic scaling, while broken lines with open circles are pseudospectral results.

lates and uses the three-center, one-electron integrals currently grows slightly faster than N^2 [see Fig. 2(b)]. A more thorough implementation of cutoffs should reduce this to N^2; further algorithmic developments, such as the multipole expansions suggested in Ref. 14, will eventually yield linear scaling for the integral assembly for sufficiently large systems.

The third section of the code, that which assembles the Fock operator in spectral space, currently scales as $\sim N^{2.75}$. This is unsurprising, since we have not yet implemented cutoffs in this part of the program. The asymptotic scaling of the Fock assembly should be N or $N \log N$; achievement of N^2 scaling for molecules the size of glutamine should be possible for most of the required operations.

While straightforward implementations of the pseudospectral and RH methods require $O(N^3)$ and $O(N^4)$ flops, respectively, cutoffs can substantially reduce the workload. As can be seen from Fig. 2(d), the overall scaling of the pseudospectral method is now $\sim N^2$ in the 80–200 basis function range; future work should lead to further reductions. The RH programs GAMESS and GAUSSIAN 86 can be seen from Table II to exhibit approximate N^4 scaling, while the CPU time for GRADSCF grows as $\sim N^{3.5}$. The scaling of these programs beyond $N = 200$ is irrelevant, since disk storage demands make such calculations impossible except for special cases. GAUSSIAN 88, when run in direct mode, achieves N^3 CPU scaling in this region. This includes, however, a rather large prefactor; GAUSSIAN 88 is the slowest of the five programs for the smaller molecules.

The scaling of the pseudospectral and RH methods be-

TABLE IV. Comparison of pseudospectral and conventional programs: Disk storage (in mW).[a]

Molecule (basis functions)	GAMESS[b]	GAUSSIAN 86	GAUSSIAN 88[b,c]	GRADSCF	Pseudospectral[c]
Cyclobutadiene (80)	5.6	6.1	0.9	5.9	4.2
Glycine (100)	20.0	21.3	1.3	13.5	5.2
Benzene (120)	27.2	28.8	1.8	27.5	6.4
Uracil (140)	47.9	50.2	2.3	48.9	7.2
Glycylglycine (175)	187.2	194.2	3.4	120.0	10.8
Glutamine (200)	314.2	325.7	4.4	d	13.4

[a–d] See notes for Table II.

TABLE V. Comparison of pseudospectral and conventional Programs: Standard vs direct SCF.[a,b]

Program	Standard SCF CPU time (s)	Direct SCF CPU time (s)	% increase
GAUSSIAN 88	374	503	34.5
Pseudospectral	59.7	68.4	14.6

[a] All data are from 6-31G** single-geometry HF calculations w/o symmetry on glycine, on a Cray X-MP EA/14se.
[b] Core memory used: Gaussian-88: 1.5 mW; pseudospectral: 1.0 mW.

yond the 200 basis function level is unclear. Ahlrichs and coworkers[31] claim $N^{2.3}$ for their RH program TURBOMOLE; this is based, however, on only two data points ($N = 450$ and 900) from calculations on a pair of highly symmetric molecules with unusual structure. We expect to have timing results for the pseudospectral program shortly for $N > 200$; this should give a better indication of asymptotic scaling behavior.

C. Disk storage and I/O

In Table IV, the disk storage requirements for each program are listed. GAMESS, GRADSCF, and GAUSSIAN 86 store two-electron integrals on disk. This requires disk storage that scales as N^4, and these programs rapidly use all available disk space. For the largest molecule tested here, glutamine, GAMESS, and GAUSSIAN 86 required over 300 megawords (Mw), or 2.4 gigabytes (Gb). The maximum storage allowed a single user on the Cray X-MP is about 4 Gb; this implies that a 250 basis function calculation (without symmetry) with either of these programs would be impossible on this machine. The program GRADSCF performs a sort in which the maximum number of elements is 200 million; this precludes calculations on glutamine or anything larger.

Both GAUSSIAN 88 and the pseudospectral code were run in "direct" mode, recalculating all necessary integrals at each iteration step. This reduces disk storage enormously, at the expense of some increase in CPU time (see Table V). The increase in CPU usage is significantly smaller for the pseudospectral method than for conventional programs, since only the medium and coarse grid integrals need to be recalculated, and since the pseudospectral integrals are not as expensive as the two-electron integrals of Eq. (1).

Table IV shows that GAUSSIAN 88 has very modest disk requirements; the pseudospectral usage is somewhat larger, but is still quite reasonable, and grows slowly with molecular size. Further, about two-thirds of this storage is due to the least-squares matrix **Q** for the finest grid. Storage of this matrix can be eliminated by constructing it immediately prior to its use, since the finest grid is used for only a single iteration.

The total data transferred to and from disk are shown in Table VI for each program. The programs which do not recalculate integrals require a huge amount of I/O, and this requirement increases rapidly with basis set size. GAUSSIAN 88 performs less I/O than the pseudospectral code for the molecules tested here; the amount of I/O performed by both programs grows as N^2.

D. Core memory

The memory used by each program is displayed in Table VII. GRADSCF adjusts the size of core memory automatically, while the GAUSSIAN programs allow the user to set the size. In general, increasing the memory for these programs decreases both CPU and I/O. Rather than try to optimize the memory-CPU-I/O combination for each molecule, we chose to run GAUSSIAN 86 and GAUSSIAN 88 with a fixed 1.5 Mw of core. This value is slightly larger than the maximum required by the pseudospectral program (1.3 Mw), and slightly smaller than that of GRADSCF (1.8 Mw). The version of GAMESS we used had a fixed memory size of 1.4 Mw.

For the molecules tested here, matrices of size N^2 (**F**, **P**, **S**, etc.) are relatively small, and the pseudospectral memory requirement is dominated by matrices which grow linearly (with a large prefactor) with basis set size. For larger problems the N^2 terms will dominate, and some minor adjustments of the code will have to be performed in order to avoid excessive memory usage in the 500–2000 basis function range.

E. Vectorization

As another measure of computational efficiency, we report in Table VIII the average number of flops performed per second for each program, for the glycine molecule. The theoretical speed of the Cray X-MP is ~210 million flops per second (210 Mflops). The Cray assembly language matrix multiply routine MXMA runs at about 200 Mflops for very large matrices; a purely scalar code can attain 12

TABLE VI. Comparison of pseudospectral and conventional programs: Total I/O (in mWords).[a]

Molecule (basis functions)	GAMESS[b]	GAUSSIAN 86	GAUSSIAN 88[b,c]	GRADSCF	Pseudo-spectral[c]
Cyclobutadiene (80)	53	74	4	83	16
Glycine (100)	266	344	7	287	21
Benzene (120)	284	385	8	440	28
Uracil (140)	593	875	15	1144	37
Glycylglycine (175)	2498	3646	22	3128	57
Glutamine (200)	3960	6133	29	(d)	82

[a–d] See notes for Table II.

TABLE VII. Comparison of pseudospectral and conventional programs: Core memory (in mWords).[a]

Molecule (basis functions)	GAMESS[b]	GAUSSIAN 88, GAUSSIAN 86[b,c]	GRADSCF	Pseudo-spectral[c]
Cyclobutadiene (80)	1.4	1.5	0.5	0.9
Glycine (100)	1.4	1.5	0.7	1.0
Benzene (120)	1.4	1.5	0.9	1.0
Uracil (140)	1.4	1.5	1.2	1.1
Glycylglycine (175)	1.4	1.5	1.8	1.2
Glutamine (200)	1.4	1.5	(d)	1.3

[a–d] See notes for Table II.

Mflops on this machine. Our code as a whole performs at a rate of 106 Mflops. GAUSSIAN 88 run in direct mode achieves 32 Mflops, due to the domination of CPU time by the integral package. Apparently this is the only part of GAUSSIAN 88 which vectorizes, since the disk-based version manages only 8.3 Mflops. GRADSCF, which was written specifically for Cray systems, runs at 17 Mflops, while the other two spectral programs tested are below 10 Mflops.

VI. CONCLUSION

In this paper we have demonstrated that the pseudospectral method for Hartree–Fock calculations is capable of reproducing the Roothaan–Hall total energies for a wide variety of molecules. Furthermore, substantial reductions in CPU usage have been achieved over the conventional programs tested here, and it has been shown that the inherent scaling advantage of the pseudospectral method, combined with cutoffs in all sections of the code, should lead to even greater timing advantages for larger molecules.

We remind the reader that the pseudospectral algorithm is not yet mature; further development will very likely reduce the CPU times listed in this paper. To illustrate this point, we note that our HF calculation on glycine in Ref. 14 took 129 CPU s; the current value (from Table II) is 68 s, a reduction of 47%.

A major goal of this project is to make possible *ab initio* quality calculations on molecular systems which are too large for conventional methods to handle. Towards this end, we expect to report soon timings for molecules in the 250–1000 basis function range. The only barrier to running such calculations is the core memory; as discussed above, some trivial reconfiguration of the code will be necessary. Installation of cutoffs in the Fock matrix assembly routines and a more thorough investigation of the convergence scheme are also planned.

Extensions of the basic pseudospectral HF theory discussed in this paper are now underway on several fronts:

1. Atomic grids and dealiasing sets are being developed for second row atoms and for transition metals.
2. The integral package is being modified to allow for the use of f and g basis functions, which will likely require upgrading of the least-squares code to handle g and perhaps h dealiasing functions.
3. The synthesis (as described in Ref. 15) of the pseudospectral program with the GVB2P5 program of Goddard and co-workers is now more complete; this work will be reported shortly, along with timings for GVB calculations.
4. A pseudospectral analytic gradient program has been completed and successfully tested; a separate paper describing this work is being prepared.[32] Additional code to compute second derivatives is under development.
5. A pseudospectral algorithm for second-order Møller–Plesset perturbation theory (MP2) has been designed and will be coded shortly.

TABLE VIII. Comparison of pseudospectral and conventional programs: Average floating point computation rates (in Mflops).[a,b]

Program	Computation rate (Mflops)
GAUSSIAN 86	5.5
GAUSSIAN 88 (disk)	8.3
GAMESS	8.5
GRADSCF	17.0
GAUSSIAN 88 (direct)	31.7
Pseudospectral (disk)	105.6
Pseudospectral (direct)	106.4

[a] All data are from 6-31G** single-geometry HF calculations w/o symmetry on glycine.
[b] GAMESS and GAUSSIAN 88 were tested on a Cray X-MP EA/14se, while all other data are from a Cray X-MP/24.

ACKNOWLEDGMENTS

This work was supported by grants from the NIH and the NSF. RAF is a Camille and Henry Dreyfus Teacher-Scholar and the recipient of a Research Career Development Award from the National Institutes of Health, Institute of General Medical Sciences. Use of the Cray X-MP/24 and X-MP EA/14se supercomputers at the University of Texas Center for High Performance Computing is gratefully acknowledged.

[1] C. C. J. Roothaan, Rev. Mod. Phys. **23**, 69 (1951); G. G. Hall, Proc. R. Soc. A**205**, 541 (1951).
[2] For flop counts see, e.g., Ref. 8, also D. Hegarty and G. Van Der Velde, Int. J. Quantum Chem. **23**, 1135 (1983).
[3] J. Almlöf, K. Faegri, Jr., and K. Korsell, J. Comput. Chem. **3**, 385 (1982).
[4] D. Cremer and J. Gauss, J. Comput. Chem. **7**, 274 (1986).
[5] M. Häser and R. Ahlrichs, J. Comput. Chem. **10**, 104 (1989).
[6] See, e.g., L. E. McMurchie and E. R. Davidson, J. Comput. Phys. **26**, 218 (1978); J. A. Pople and W. J. Hehre, *ibid.* **27**, 161 (1978); J. Rys, M. Dupuis, and H. F. King, J. Comput. Chem. **4**, 154 (1983).

[7] S. Obara and A. Saika, J. Chem. Phys. **84**, 3963 (1986).

[8] M. Head-Gordon and J. A. Pople, J. Chem. Phys. **89**, 5777 (1988).

[9] P. W. Fowler, P. Lazzeretti, and R. Zanasi, Chem. Phys. Lett. **165**, 79 (1990).

[10] R. A. Friesner, Chem. Phys. Lett. **116**, 39 (1985).

[11] R. A. Friesner, J. Chem. Phys. **85**, 1462 (1986).

[12] R. A. Friesner, J. Chem. Phys. **86**, 3522 (1987).

[13] R. A. Friesner, J. Phys. Chem. **92**, 3091 (1988).

[14] M. N. Ringnalda, Y. Won, and R. A. Friesner, J. Chem. Phys. **92**, 1163 (1990).

[15] J.-M. Langlois, R. P. Muller, T. R. Coley, W. A. Goddard, III, M. N. Ringnalda, Y. Won, and R. A. Friesner, J. Chem. Phys. **92**, 7488 (1990).

[16] S. A. Orszag, Stud. Appl. Math. 51, 253 (1972); S. A. Orszag, Stud. Appl. Math. **50**, 293 (1971); S. A. Orszag, Phys. Rev. Lett. **26**, 1100 (1971); D. Gottlieb and S. Orszag, *Numerical Analysis of Spectral Methods; Theory and Application* (SIAM, Philadelphia, 1977).

[17] W. J. Hehre, R. Ditchfield, and J. A. Pople, J. Chem. Phys. **56**, 2257 (1972); P. C. Harihan and J. A. Pople, Theor. Chim. Acta (Berlin) **28**, 213 (1973).

[18] F. W. Bobrowicz and W. A. Goddard, III, in *Modern Theoretical Chemistry: Methods of Electronic Structure Theory*, edited by H. F. Schaefer, (Plenum, New York, 1977), Vol. 3, p. 79.

[19] V. I. Lebedev, Zh. vychisl. Mat. mat. Fiz. **15**, 48 (1975).

[20] V. I. Lebedev, Zh. vychisl. Mat. mat. Fiz. **16**, 293 (1976).

[21] V. I. Lebedev, Sibersk. Mat. Zh. **18**, 132 (1977).

[22] A. H. Stroud, *Approximate Calculation of Multiple Integrals* (Prentice-Hall, Englewood Cliffs, New Jersey 1971).

[23] G. A. Glatzmaier, J. Comput. Phys. **55**, 461 (1984).

[24] H. Taketa, S. Huzinaga, and K. O-Ohata, J. Phys. Soc. Jpn. **21**, 2313 (1966).

[25] I. Shavitt, in *Methods in Computational Physics*, edited by B. Alder, S. Fernbach, and M. Rotenberg. (Academic, New York, 1963), Vol. 2, p. 1.

[26] W. J. Hehre, L. Radom, P. v. R. Schleyer, and J. A. Pople, *Ab Initio Molecular Orbital Theory* (Wiley, New York, 1986).

[27] K. Siam, V. J. Klimkowski, J. D. Ewbank, C. Van Alsenoy, and L. Schäfer, J. Mol. Struct. Theochem. **110**, 171 (1984).

[28] L. Harsányi, P. Császár, A. Császár, and J. E. Boggs, Int. J. Quantum Chem. **29**, 799 (1986).

[29] A. Kvick, A. R. Al-Karaghouli, and T. F. Koetzle, Acta Cryst. B **33**, 3796 (1977).

[30] T. F. Koetzle, M. N. Frey, M. S. Lehmann, and W. C. Hamilton, Acta Cryst. B **29**, 2571 (1973).

[31] R. Alrichs, M. Bär, M. Häser, H. Horn, and C. Kölmel, Chem. Phys. Lett. **162**, 165 (1989).

[32] Y. Won, J.-G. Lee, M. N. Ringnalda, and R. A. Friesner, Chem. Phys. Lett. (submitted).