Computing molecular electrostatic potentials with the PRISM algorithm

Benny G. Johnson [1], Peter M.W. Gill, John A. Pople
Department of Chemistry, Carnegie Mellon University, Pittsburgh, PA 15213, USA

and

Douglas J. Fox
Gaussian, Inc., 4415 Fifth Avenue, Pittsburgh, PA 15213, USA

Received 31 December 1992; in final form 15 February 1993

The PRISM integral algorithm has been applied to the computation of the ab initio molecular electrostatic potential and its derivatives. Implementational details which are relevant to the additional efficiency of the algorithm in the electrostatic case are discussed. On a range of machines, CPU timings of the PRISM electrostatic properties program, which is included in the GAUSSIAN 92 quantum chemistry package, reveal a dramatic performance increase (in some cases more than two orders of magnitude) over other commonly used electrostatic programs (GAUSSIAN 90, GAMESS, MOPAC ESP, CHELPG). In addition, timings are reported for a particularly large electrostatic potential evaluation job on the six base-pair oligonucleotide CTCGAG ($C_{116}H_{138}N_{46}O_{68}P_{10}^{10-}$).

1. Introduction

The molecular electrostatic potential has long been recognized as a useful tool in the study of how a molecular system interacts with its surroundings. It has found application in varied areas of research [1], including, but not limited to, molecular reactivity [2–5] and determination of net atomic charges [6–14]. However, despite its utility, many of the available programs for computing the potential are severely inefficient. For example, potential evaluation by some programs can often take longer than deriving a Hartree–Fock wavefunction from which the potential is computed, a decidedly unbalanced state of affairs.

The electrostatic potential $V(\boldsymbol{r})$ is rigorously defined as a quantum mechanical expectation value, and is given in the ab initio LCAO framework by

$$V(\boldsymbol{r}) = \sum_A \frac{Z_A}{|\boldsymbol{r}-\boldsymbol{R}_A|} - \sum_{\mu\nu} P_{\mu\nu} \int \frac{\phi_\mu(\boldsymbol{r}_1)\phi_\nu(\boldsymbol{r}_1)}{|\boldsymbol{r}-\boldsymbol{r}_1|}\,\mathrm{d}\boldsymbol{r}_1\,, \tag{1}$$

where Z_A is the nuclear charge of atom A centered at $\boldsymbol{R}_A$, ϕ_μ and ϕ_ν are orbital basis functions, and $P_{\mu\nu}$ is the corresponding element of an appropriate density matrix. The first summation is the classical contribution to the electrostatic potential of the atomic nuclei, treated as point charges, and is of course trivial to evaluate. The second summation gives the contribution of the electronic charge distribution, which is nontrivial due to the three-center one-electron integrals over basis functions required.

The evaluation of these integrals is the rate-limiting step in an electrostatic potential calculation, and it is their efficient computation where the basis functions are of the Gaussian type which is the focus of this Letter. We have applied the new PRISM integral algorithm [15–20] to this problem, with impressive

[1] To whom correspondence should be addressed. E-mail: johnson@cmchem.chem.cmu.edu

results. In the remainder of this Letter, we first describe the particularization of the PRISM algorithm to electrostatic potential evaluation, and then compare the practical performance of our implementation with several other electrostatic programs commonly used. CPU timings demonstrate substantial performance improvement over the previous methods.

2. Computational method

It is not our intent to provide a comprehensive review of the PRISM algorithm, only a brief summary before discussing details relevant to the electrostatic problem. For a complete description, and for definitions of terms and notations, the reader is directed to refs. [15–20]. PRISM is a methodology for the computation of general quantities called *brakets*, which are two-electron inner-product functionals of contracted Gaussians. Particular examples of brakets include two-electron repulsion integrals (ERIs) and their arbitrary-order derivatives, and PRISM has already been demonstrated to be quite efficient in the computation of these [18,20]. PRISM can be characterized as a synthesis of generalizations of the McMurchie–Davidson integral method [21], and of Head-Gordon and Pople's improvement [22] upon the Obara–Saika method [23]. The major strength of PRISM is its ability to perform contraction of the brakets at the point of maximum computational efficiency for given angular momentum and degree of contraction, which is a degree of flexibility not possible with other algorithms. This is accomplished through use of a set of general recurrence relations which provide multiple computational routes, called "paths", from an initial set of auxiliary integrals with zero angular momentum to the desired brakets. These paths may be diagrammed on the surfaces of rectangular prisms, lending the algorithm its name.

The first step in computing electrostatic potentials with PRISM is to express the requisite integrals as brakets. This can be accomplished by considering the electrostatic integrals in eq. (1) as the Coulomb interaction of the overlap distribution $\phi_\mu(\boldsymbol{r}_1)\phi_\nu(\boldsymbol{r}_1)$ (electron 1) with the distribution $\delta(\boldsymbol{r}_2-\boldsymbol{r})$ (electron 2). The bra may simply be taken as the product $\phi_\mu(\boldsymbol{r}_1)\phi_\nu(\boldsymbol{r}_1)$, while the delta-function is phrased as a ket by taking it as the product of two uncontracted s-type Gaussians centered at $\boldsymbol{r}$: a normalized function with infinite exponent, and an unnormalized function with zero exponent. In the notation of ref. [16], we set $K_C=K_D=1$, $\gamma=\infty$, $\delta=0$, $\boldsymbol{C}=\boldsymbol{D}=\boldsymbol{r}$, and $U_Q=1$.

It should be noted that, since the electrostatic potential depends upon integrals which are brakets, the arbitrary-order derivatives of the potential with respect to the components of the position vector $\boldsymbol{r}$ (electric field, electric field gradient, etc.) are also composed of brakets and hence can be evaluated by PRISM. Our implementation includes the capability to evaluate these derivatives, but for the purpose of illustrating the use of PRISM to compute electrostatic properties, it suffices to restrict the discussion to the potential only, and we therefore do so.

Before commencing integral evaluation, a preliminary pass over contracted shell-pairs is executed, in order to discard primitive products which yield a negligible contribution and to compile useful data on the significant shell-pair components, in the same spirit as when computing ERIs. The implementation of this phase is extremely important because it determines the scaling of the integral cost with the size of the molecular system, regardless of the efficiency of the integral algorithm itself. An algorithm which computes *all* integrals in eq. (1) requires $\mathcal{O}(N^2)$ operations per potential evaluation, where N is the number of basis functions, regardless of how efficiently these integrals are worked out. However, for large systems the value of most integrals will be essentially zero due to insufficient basis function overlap, and asymptotically there are only $\mathcal{O}(N)$ which are significant. As the most efficient way to deal with an integral is to avoid it if at all possible, this problem has been carefully addressed in the general PRISM algorithm.

In the electrostatic case, it is possible to develop stronger criteria for accepting or rejecting a shell-pair than used in the PRISM ERI program, as the potential is a one-electron (shell-pair) quantity, while ERIs are two-electron (shell-quartet) quantities. The cutoff parameter

$$V^*_{ab}=P^*_{ab}W^*_{ab} \qquad (2)$$

is computed for each primitive shell-pair $[ab|$, where P^*_{ab} is the maximum density matrix element corresponding to the shell-pair, and W^*_{ab} is the maximum absolute potential of all the basis-function pairs in the shell-pair, which is found either by an exact or approximate method. This is then compared with a threshold value ϵ/s, where ϵ is the desired accuracy and s is a "safety factor" to account for error propagation of neglected integrals and the use of approximations to W^*_{ab}. We have found $s=5$ to be sufficient. The primitive shell-pair is rejected if its V^*_{ab} value is below the threshold. (Incidentally, we have also applied this cutoff criterion in the context of ERI evaluation [24].)

In our implementation, exact values of W^*_{ab} are used for total angular momentum zero and one. This corresponds in the case of zero angular momentum to the bound previously given by Gadre et al. [25]. For computational simplicity, an approximation to W^*_{ab} is used for higher angular momentum. The primitive basis function product with all angular momentum in one Cartesian direction is expressed as a sum of Hermite Gaussian functions (see eq. (45) of ref. [16]), and the sum of the exact W^*_{ab} values is taken for only the s-type terms (corresponding to the leading terms in a multipole expansion). In the case of odd total angular momentum, the leading p-type term is also retained as there will be no s-type terms in the case where shells a and b are concentric.

After the significant shell-pair data have been compiled, integral evaluation begins. The general philosophy of PRISM is to compute batches of integrals (not to be confused with classes of integrals [22]) having the same angular momentum and contraction characteristics together, in order to make the best use of intermediate quantities and to maximize vector performance [15]. In the present context, an improvement upon this approach is possible which merits elaboration.

The code was written on the assumption that the value of the electrostatic potential will in general be desired on a grid of a large number of points. This will be the case, for example, when mapping out the features of the potential for a reactivity study, or when deriving net atomic charges from the potential. The program is therefore structured to vectorize over batches of grid points, which are the identical delta-function kets differing only in position in space. Instead of selecting as many shell-pairs as possible with the appropriate characteristics for bras to be paired with the kets, however, only *one* such shell-pair is chosen at a time, and is paired with all kets to form the integral batches. This strategy, referred to as the "fixed shell-pair" scheme, affords considerable additional computational efficiency, since the bra shell-pair data are constant for all members of the batch (and trivially likewise for all ket data except position). Furthermore, this simplification is achieved without loss of vector performance when the number of grid points is large.

Initially, the fixed shell-pair scheme greatly simplifies computation of the elementary integrals denoted $[\mathbf{0}]^{(m)}$ [17], as their dependence upon the necessary "shell-quartet" quantities has been reduced to merely the distance between the grid point and the primitive shell-pair center. The pertinent formula [16] is

$$[\mathbf{0}]^{(m)} = U(2\theta^2)^{m+1/2}(2/\pi)^{1/2}F_m(t)\,, \qquad (3)$$

where $F_m(t)$ is the well-known auxiliary function, and the preceding factors depend here solely on bra shell-pair data. Therefore, the value of this prefactor need be evaluated *only once*, with the result being applied in computing all members of the batch. This results in a computational saving both of floating-point operations (flops), and more importantly, of memory operations (mops) [26], as well as a saving of vectors of storage for the values of U and $2\theta^2$. Such savings are examples of general ramifications of the fixed shell-pair scheme in all phases of the integral evaluation procedure, as will be demonstrated.

At this point, the PRISM path which is the most efficient for the integral class at hand is selected. In the case of electrostatic integrals, many sets of formally distinct paths become equivalent, because neither the ket-contraction nor the ket-transformation step is required. To illustrate the path-dependent economizations which are possible, we consider the CCTTT path, which is also discussed in ref. [18].

This path begins by bra-contraction of the $[\mathbf{0}]^{(m)}$ integrals (ket-contraction is absent here). The usefulness of simultaneously incorporating scalings of exponent ratios has previously been noted [16]. The scaled contraction formula is

$$Z_j = \sum_{k=1}^{K} S_{jk} W_{jk}, \tag{4}$$

where Z_j is the jth contracted integral in the batch, K is the bra degree of contraction, and the S_{jk} and W_{jk} are, respectively, the scalings and primitive integrals. This is implemented with the innermost loop running over the members of the batch and the outer loop over primitives unrolled, and thanks to the fixed shell-pair scheme, the scalings become invariant with respect to the batch loop index j, i.e. eq. (4) becomes

$$Z_j = \sum_{k=1}^{K} S_k W_{jk}. \tag{5}$$

Therefore, only one set of K scalings need be loaded per batch, rather than one set per innermost loop iteration. This is seen to reduce the number of mops involved in the contraction by nearly half, a substantial savings. Additionally, the storage required for the primitive scalings becomes negligible.

The contracted integrals $(\mathbf{0})^{(m)}$ are next transformed to scaled contracted one-center Hermite integrals by eq. (22) of ref. [18], derived from the familiar identity for uncontracted integrals due to McMurchie and Davidson [21]. The recurrence relation has the abstract form

$$Z = (B_i - A_i) W_1 + (C_i - D_i) W_2 + (D_i - B_i) W_3 - m W_4, \tag{6}$$

where W_1–W_4 are intermediate integrals, m is a constant, and the batch index subscripts have been omitted. The relevant items in the above formula are the intercenter distance components $(B_i - A_i)$ and $(C_i - D_i)$. The first pertains to the bra shell-pair intercenter distance, and is therefore invariant in the innermost loop over batch members due to the fixed shell-pair scheme. This eliminates one mop per iteration from the loop, and frees the storage otherwise needed for the different $(B_i - A_i)$ values. Additionally, a preliminary test on the magnitude of $(B_i - A_i)$ may be performed before entering the loop over batch members, and if it is zero, a special-case coding of eq. (6) with the first term omitted may be executed instead. This test, which can save an additional mop and two flops per batch member, could not be implemented without the fixed shell-pair scheme because it would have to be executed inside the batch loop, sacrificing vector performance. The second quantity $(C_i - D_i)$ is a ket shell-pair intercenter distance component, which is zero, thereby dropping the second term from the formula altogether. The quantity $(D_i - B_i)$ depends on the distance from a grid point to one of the bra centers, and hence is not constant in the batch. The overall result is that the 8 mops and 7 flops required per iteration in the most general case are reduced to 5 mops and 5 flops (4 mops and 3 flops when $(B_i - A_i)$ is zero), along with the associated memory savings.

The final step on this path is the bra-transformation, since no ket-transformation is required. This is accomplished with eq. (45) in ref. [16], which has the form

$$Z = p_i W_1 + W_2 + (B_i - A_i) W_3, \tag{7}$$

where W_1–W_3 are intermediate integrals and p_i is a constant. As with eq. (6), the intercenter distance component $(B_i - A_i)$ becomes a scalar in the fixed shell-pair scheme, allowing the same implementational efficiences discussed for $(B_i - A_i)$ above.

3. Practical performance

From the preceding section, PRISM is expected from a theoretical standpoint to be quite efficient in computing electrostatic integrals. However, the most pragmatic test of the merit of an integral algorithm is its performance in actual implementation on real systems. Therefore, a head-to-head comparison was performed with the following five electrostatic programs:

(1) Link 602 of GAUSSIAN 90 [27], which uses the Rys quadrature integral method [28].

(2) The GAMESS electrostatic program [29], also based on Rys.

(3) MOPAC ESP [30], which is based on the Obara–Saika integral method [23].

(4) A vectorized version [31] of the CHELPG program [11,13], which uses explicit special-case integral formulae.

(5) Link 602 of GAUSSIAN 92 [32], which uses the current method.

Electrostatic potentials were computed by each program for two molecular systems: naphthalene, which is relatively small, and phenylalanine, which

is about twice as large. The grids of points for these were generated by the Singh and Kollman algorithm [10] for fitting of potential-derived atomic charges. Hartree–Fock wavefunctions were calculated and used for potential evaluation by all programs except the semiempirical package MOPAC ESP; here the MNDO method was used instead. However, this program projects the semiempirical wavefunction onto a standard orbital basis set and uses the resulting density matrix to compute the potential, and so timing comparisons between MOPAC ESP and the other programs are valid, even though the actual numerical results will differ. Both the STO-3G and 6-31G* basis sets were used, to give a variety in the angular momentum and degree of contraction of the integrals computed. (MOPAC ESP does not support 6-31G*, so only STO-3G was used there.) As the particular grid method used here does not retain the molecular point-group symmetry, symmetry cannot be exploited in any of these calculations; however, G92 does have the capability to use full abelian symmetry with symmetrically oriented regular grids.

Program defaults for integral accuracy were used in all cases. The G92 Link 602 default is 10^{-6} au, which is sufficient to give around 10^{-4} to 10^{-5} accuracy in the potential values. The defaults of the other programs are all tighter than this, which is overconservative when the objective of the calculation is, e.g., visualization of the electrostatic potential or determination of potential-derived charges, tasks for which highly accurate potential values are not required and the extra computational effort is therefore wasted. Agreement to the above degree of the potentials computed by the various programs was explicitly verified (except, of course, for MOPAC ESP).

The above calculations were carried out with as many of the five programs as were available on each of the following three platforms: VAX station 3100, IBM RS/6000 320 and Cray Y-MP/8-32. The total CPU times for potential evaluation are presented in tables 1–3, and will now be discussed in turn.

Table 1 gives timings on a VAX station 3100 for G90, MOPAC ESP and G92, with times relative to G92 given in parentheses. Since this is a scalar sequential platform, performance on this machine is essentially determined simply by the number of mops and flops executed by the programs, so comparisons between the programs can be made which are independent of vector effects. First of all, it is readily apparent that G90 is drastically inferior to the other two programs on the VAX. This is due to the use of a less sophisticated underlying integral algorithm (Rys), exacerbated by a poor implementation of that algorithm and an ineffectual cutoff scheme. Although G92 is the clear winner of the three, the MOPAC ESP timings are also very respectable compared to G90, and at the time of its introduction this program represented a dramatic step forward in electrostatic potential codes.

The relative speedup of G92 compared to G90 is greater for STO-3G than for 6-31G*. This is because the Rys method handles high contraction in the basis set poorly (the primitive integrals are simply added

Table 1
VAX station 3100 CPU timings for electrostatic potential calculations by various programs

Molecule	Grid points	Basis set	Basis functions	CPU time [a] (s)		
				G90	MOPAC ESP [b]	G92
naphthalene	916	STO-3G	58	1840 (17.5)	425 (4.05)	105
		6-31G*	166	5430 (14.3)	– –	379
phenylalanine	1278	STO-3G	96	6980 (24.0)	1540 (5.29)	291
		6-31G*	272	20900 (19.2)	– –	1090

[a] Values in parentheses are relative to G92 time.
[b] The 6-31G* basis set is not available with MOPAC ESP.

together at the end), while performing better with weakly contracted, high angular momentum basis sets like 6-31G*.

One of the most important conclusions which can be made from table 1 is that the relative speedup of G92 over both the other programs increases as the size of the molecular system increases. G92 completes the naphthalene STO-3G calculation 17.5 times faster than G90 and 4.05 times faster than MOPAC ESP; however, for phenylalanine STO-3G the G92 speedups increase to 24.0 and 5.29, respectively. This shows that the G92 cutoff scheme is more effective than in the other two programs (if they had the same efficiency the relative speedup would remain constant upon going to a bigger molecule), and that the MOPAC ESP cutoffs are better than in G90. The G90 timings actually show that the cost per grid point is almost fully quadratic in the size of the basis set, indicating that virtually no insignificant integrals are being successfully prescreened. This is a clear illustration of the importance of having a good cutoff procedure, and has obvious implications for calculations on very large systems.

Table 2 gives timings on an IBM RS/6000 320, a popular RISC-based workstation. Only G90 and G92 were available to us on this platform. As on the VAX, G90 offers no competition to G92, with the same general trends being observed. It is worth noting that the G92 relative speedups are considerably greater here than on the VAX; this is related to the fact that G92 vectorizes much better than G90.

The third machine used in this study was the Cray Y-MP/8-32, a large vector supercomputer which is typical of the sort of platform in mind when the PRISM algorithm was designed. All five programs were run on this machine using a single processor, with the results compiled in table 3. The first two, G90 and GAMESS, are two orders of magnitude slower than G92 on these systems and exhibit no vectorization whatsoever. The timing ratios of MOPAC ESP to G92 are about six times as great as on the VAX, which is attributed to the comparative vectorizability of the two programs. MOPAC ESP mainly vectorizes over the primitive basis functions, which offers only limited vector performance compared to vectorizing over the grid points as is done in G92.

Next is the vectorized CHELPG program, a version specially written for Y-MP in order to improve the performance of the original CHELPG, which does not vectorize. For small systems this program is quite fast and achieves good vectorization, coming within a factor of three of G92 for naphthalene STO-3G. The explicitly coded formulae for d-containing integrals are not efficient, and a marked degradation in performance is observed when moving to 6-31G*. As in G92, vectorization is over grid points, but full quadratic storage must be allocated for each point, which decreases the maximum vector length within a fixed amount of memory as the basis set size is increased. For example, a vector length of 512 was easily possible for naphthalene STO-3G within the four megawords of memory used, but for phenylalanine 6-31G* only a vector length of 64 could be obtained, impairing performance. This is to be contrasted with the fixed shell-pair scheme in G92, which requires

Table 2
IBM RS/6000 320 CPU timings for electrostatic potential calculations by various programs

Molecule	Grid points	Basis set	Basis functions	CPU time [a] (s)	
				G90	G92
naphthalene	916	STO-3G	58	355 (26.3)	13.5
		6-31G*	166	1060 (24.1)	44.0
phenylalanine	1278	STO-3G	96	1380 (37.7)	36.6
		6-31G*	272	4080 (34.0)	120

[a] Values in parentheses are relative to G92 time.

Table 3
Cray Y-MP/8-32 CPU timings for electrostatic potential calculations by various programs

Molecule	Grid points	Basis set	Basis functions	CPU time [a,b] (s)				
				G90	GAMESS	MOPAC ESP [c]	CHELPG	G92
naphthalene	916	STO-3G	58	137	62.8	14.0	1.95	0.606
				(226)	(104)	(23.1)	(3.22)	
		6-31G*	166	411	190	–	8.90	2.11
				(195)	(90.0)	–	(4.23)	
phenylalanine	1278	STO-3G	96	544	217	48.4	7.27	1.51
				(360)	(144)	(32.1)	(4.81)	
		6-31G*	272	2370	623	–	42.4	5.47
				(433)	(114)	–	(7.75)	
CTCGAG ($C_{116}H_{138}N_{46}O_{68}P_{10}^{10-}$)	9310	STO-3G	1378					163

[a] One processor, vector mode.
[b] Values in parentheses are relative to G92 time.
[c] The 6-31G* basis set is not available with MOPAC ESP.

extremely little storage for basis set information for each batch of integrals, maximizing the vector length. All G92 calculations on the Y-MP had a vector length equal to the total number of grid points, ensuring essentially perfect vectorization. However, due to its memory restrictions, the scope of CHELPG is limited to small and medium-sized systems. It should be noted that the drawbacks mentioned above mainly result from the structure of the original program, which imposed constraints on the subsequent approach to vectorizing the code. As this program is the second fastest of the five in spite of this, the effort put into this code has been quite successful.

As mentioned earlier, G92 is able to obtain high vector performance with little difficulty. The dominant subroutines in terms of CPU time are Calc0G, which generated the initial $[\mathbf{0}]^{(m)}$ integrals, and DoConG, which performs the contraction. These together account for around 55% of the total CPU time, and both run at speeds of 175 Mflops or more on a single processor. No other individual routine accounts for more than 10% of the total CPU time.

Finally, the last row of table 3 gives the timing of an extremely large electrostatic potential job. The HF/STO-3G potential of the oligonucleotide CTCGAG (molecular formula $C_{116}H_{138}N_{46}O_{68}P_{10}^{10-}$, 1378 basis functions) was calculated on a Singh–Kollman grid of 9310 points in 163 seconds. This is totally insignificant compared with approximately 11 h of CPU time required for the SCF calculation. The vector length was the full 9310 points. Comparison with the corresponding time for phenylalanine shows that the cost per grid point is indeed essentially linear in system size. The undertaking of this job would absolutely not be advised using each of the other four programs; MOPAC ESP and CHELPG due to memory requirements, and G90 and GAMESS due to extremely poor performance. We estimate this calculation by G90 would require approximately nine Y-MP CPU days.

4. Conclusion

The PRISM algorithm, originally designed for the efficient computation of ERIs and their derivatives, is also very well suited to the computation of the electrostatic potential and its derivatives. Our implementation, which is included in GAUSSIAN 92, has been shown to perform extremely well on a variety of machines, and to be vastly superior to the four other electrostatic programs studied. It is now possible to evaluate the potential on large grids in a very small fraction of the time required to solve the HF SCF equations, dramatically illustrated by the under-three-minute CTCGAG potential calculation on the Y-MP. One especially rewarding area of application for this method would be the calculation of

semiempirical electrostatic potentials of extremely large systems, as the SCF time is negligible compared with ab initio methods and most of the CPU time is spent in the electrostatic calculation.

Acknowledgement

BGJ gratefully acknowledges the contributions of the following persons: Ken Merz for many stimulating discussions and for providing a copy of the MOPAC ESP program, Carlos Gonzalez for assistance with the CTCGAG calculation on the Y-MP, Kevin Moore for providing a copy of the vectorized CHELPG program, George Fitzgerald for performing the Y-MP MOPAC ESP timings, and Mike Frisch for providing a G92 checkpoint file for CTCGAG. This work was supported by Cray Research, Inc. and the Pittsburgh Supercomputing Center under Grant CHE910042P, and by the National Science Foundation under Grant CHEM-8918623. BGJ thanks the Mellon College of Science for a Graduate Fellowship. An account of this work was presented at the 32nd annual Sanibel Symposium held in St. Augustine, Florida.

References

[1] P. Politzer and D.G. Truhlar, eds. Chemical applications of atomic and molecular electrostatic potentials (Plenum Press, New York, 1981).
[2] E. Scrocco and J. Tomasi, Topics Current Chem. 42 (1973) 95.
[3] A. Goursot, F. Fajula, C. Daul and J. Weber, J. Phys. Chem. 92 (1988) 4456.
[4] J. Murray and P. Politzer, Chem. Phys. Letters 152 (1988) 364.
[5] D. Dehareng, G. Dive and J.M. Ghuysen, Theoret. Chim. Acta 79 (1991) 141.
[6] F.A. Momany, J. Phys. Chem. 82 (1978) 592.
[7] E. Scrocco and J. Tomasi, Advan. Quantum Chem. 11 (1978) 115.
[8] P.H. Smit, J.L. Derissen and F.B. van Duijneveldt, Mol. Phys. 37 (1979) 521.
[9] S.R. Cox and D.E. Williams, J. Comput. Chem. 2 (1981) 304.
[10] U.C. Singh and P.A. Kollman, J. Comput. Chem. 5 (1984) 129.
[11] L.E. Chirlain and M.M. Francl, J. Comput. Chem. 8 (1987) 894.
[12] R.J. Woods, M. Khalil, W. Pell, S.H. Moffat and V.H. Smith Jr., J. Comput. Chem. 11 (1990) 297.
[13] C.M. Breneman and K.B. Wiberg, J. Comput. Chem. 11 (1990) 361.
[14] B.H. Besler, K.M. Merz Jr. and P.A. Kollman, J. Comput. Chem. 11 (1990) 431.
[15] P.M.W. Gill, M. Head-Gordon and J.A. Pople, Intern. J. Quantum Chem. Symp. 23 (1989) 269.
[16] P.M.W. Gill, M. Head-Gordon and J.A. Pople, J. Phys. Chem. 94 (1990) 5564.
[17] P.M.W. Gill, B.G. Johnson and J.A. Pople, Intern. J. Quantum Chem. 40 (1991) 745.
[18] P.M.W. Gill and J.A. Pople, Intern. J. Quantum Chem. 40 (1991) 753.
[19] B.G. Johnson, P.M.W. Gill and J.A. Pople, Intern. J. Quantum Chem. 40 (1991) 809.
[20] P.M.W. Gill, Advan. Quantum Chem., in press.
[21] L.E. McMurchie and E.R. Davidson, J. Comput. Phys. 26 (1978) 218.
[22] M. Head-Gordon and J.A. Pople, J. Chem. Phys. 89 (1988) 5777.
[23] S. Obara and A. Saika, J. Chem. Phys. 84 (1986) 3963.
[24] P.M.W. Gill, B.G. Johnson and J.A. Pople, to be published.
[25] S.R. Gadre, S.A. Kulkarni and R.K. Pathak, J. Chem. Phys. 91 (1989) 3596.
[26] M.J. Frisch, B.G. Johnson, P.M.W. Gill, D.J. Fox and R.H. Nobes, Chem. Phys. Letters 206 (1993) 225.
[27] M.J. Frisch, M. Head-Gordon, G.W. Trucks, J.B. Foresman, H.B. Schlegel, K. Raghavachari, M.A. Robb, J.S. Binkley, C. Gonzalez, D.J. DeFrees, D.J. Fox, R.A. Whiteside, R. Seeger, C.F. Melius, J. Baker, R.L. Martin, L.R. Kahn, J.J.P. Stewart, S. Topiol and J.A. Pople, GAUSSIAN 90 (Gaussian, Inc., Pittsburgh, PA, 1990).
[28] M. Dupuis, J. Rys and H.F. King, J. Chem. Phys. 65 (1976) 111;
H.F. King and M. Dupuis, J. Comput. Phys. 21 (1976) 144;
J. Rys, M. Dupuis and H.F. King, J. Comput. Chem. 4 (1983) 154.
[29] M.W. Schmidt, K.K. Baldridge, J.A. Boatz, J.H. Jensen, S. Koseki, M.S. Gordon, K.A. Nguyen, T.L. Windus and S.T. Elbert, QCPE Bulletin 10 (1990) August issue.
[30] B.H. Besler and K.M. Merz Jr., program obtained by anonymous FTP from retina.chem.psu.edu
[31] K. Moore and L. Balbes, program obtained by anonymous FTP from duck.ncsc.org
[32] M.J. Frisch, G.W. Trucks, M. Head-Gordon, P.M.W. Gill, M.W. Wong, J.B. Foresman, B.G. Johnson, H.B. Schlegel, M.A. Robb, E.S. Replogle, R. Gomperts, J.L. Andres, K. Raghavachari, J.S. Binkley, C. Gonzalez, R.L. Martin, D.J. Fox, D.J. DeFrees, J. Baker, J.J.P. Stewart and J.A. Pople, GAUSSIAN 92 (Gaussian, Inc., Pittsburgh, PA, 1992).