RESEARCH ARTICLE | NOVEMBER 01 1988

A method for two-electron Gaussian integral and integral derivative evaluation using recurrence relations

Martin Head-Gordon; John A. Pople

J. Chem. Phys. 89, 5777–5786 (1988)
https://doi.org/10.1063/1.455553

A method for two-electron Gaussian integral and integral derivative evaluation using recurrence relations

Martin Head-Gordon and John A. Pople
Department of Chemistry, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213

(Received 26 May 1988; accepted 20 July 1988)

An efficient method is presented for evaluating two-electron Cartesian Gaussian integrals, and their first derivatives with respect to nuclear coordinates. It is based on the recurrence relation (RR) of Obara and Saika [J. Chem. Phys. **84**, 3963 (1986)], and an additional new RR, which are combined together in a general algorithm applicable to any angular momenta. This algorithm exploits the fact that the new RR can be applied outside contraction loops. It is shown, by floating point operation counts and comparative timings, to be generally superior to existing methods, particularly for basis sets containing *d* functions.

I. INTRODUCTION

In the popular *ab initio* Hartree–Fock (HF)[1] method for studying molecular electronic structure, the rate-limiting step is the generation and handling of the infamous two-electron repulsion integrals (ERIs), and their derivatives. This is because the number of ERIs increases as the fourth power of the size of the problem (i.e., N^4, where N is the number of basis functions), while other computations in the HF method rise as no more than N^3. The ERIs are necessary to form the matrix of the HF Hamiltonian (Fock matrix) while derivatives are needed to obtain the gradient of the energy with respect to nuclear coordinates.[2] For very large HF calculations, there may not be enough disk space to store the ERIs, and it is then necessary to recompute them whenever they are needed. For example, recomputing ERIs on each iteration of the self-consistent field (SCF) procedure constitutes the direct SCF method.[3] Analogously, recomputing ERIs on each cycle of the iterative coupled perturbed HF procedure[4] is necessary to permit vary large harmonic frequency calculations. The viability of these schemes, as well as more conventional HF calculations, depends directly on the availability of highly efficient methods for the evaluation of ERIs and their derivatives.

Almost all efficient algorithms for ERI calculation are based upon the use of Gaussian basis functions, originally suggested by Boys.[5] A number of comprehensive reviews on methods for integral evaluation over Gaussian basis functions have appeared,[6–8] covering developments in the field prior to 1986, when a particularly promising method was reported by Obara and Saika (OS).[9] OS derived a recurrence relation (RR) which relates a given ERI to other integrals, both true ERIs and "auxiliary" ERIs, of lower angular momentum (this RR is also implicit[10] in the earlier method of Schlegel[11] for derivative integrals). By repeated application of the RR, an ERI may be reduced to integrals involving only zero angular momentum *s* functions, which can be readily evaluated by standard methods.[9] The RR can often be applied in many different sequences, particularly if *d* functions or higher are involved, and it is in principle a tree-search problem to find the most efficient sequence for a given ERI type.

The algorithm used by OS to apply the RR, although not precisely specified in their paper, was fractionally faster than the Pople–Hehre (PH) method[12] for the test cases reported. The PH method had hitherto been considered most efficient for ERIs involving only *s* and *p* functions. The OS method was more than four times faster than an implementation[13] of the Rys polynomial method[14] for examples involving *s* and *p* functions, and was about three times faster for basis sets containing *s*, *p*, and *d* functions, on the computer they used. This latter result is especially significant since the Rys method has generally been the method of choice for ERIs involving *d* or *f* functions. For example, in the GAUSSIAN 86 system of programs,[15] it is used for this purpose in conjunction with the PH method for *s* and *p* functions. Finally, for first derivatives of ERIs, the OS method was again several times faster than a Rys polynomial based alternative,[13] although the basis set used in their example involved just *s* and *p* functions, for which the method of Schlegel[11] is also several times faster than the Rys method.[16]

These encouraging results provide the main motivation for this work, which is an alternative implementation of the OS RR, based upon a simple and general algorithm. In Sec. II, the necessary theory is given, consisting of a review of the OS RR, and the introduction of a second RR which is also crucial to our approach. The proposed algorithm for ERI evaluation is described in detail in Sec. III, followed by the extensions necessary to obtain first derivatives in Sec. IV. Finally, the performance of the new method for integrals (Sec. V) and derivatives (Sec. VI) is assessed by floating point operation counts and timing comparisons with existing GAUSSIAN 86 codes.

II. RECURRENCE RELATIONS FOR ERI EVALUATION

An unnormalized primitive Cartesian Gaussian function centered at **A** with exponent α_k is

$$\varphi_{ak}(\mathbf{r}) = (x-A_x)^{a_x}(y-A_y)^{a_y}(z-A_z)^{a_z} \times \exp[-\alpha_k(\mathbf{r}-\mathbf{A})^2], \quad (1)$$

where

$$\mathbf{a} = (a_x, a_y, a_z) \quad (2)$$

is a set of three integers, the sum of which is the angular momentum of the Gaussian. Thus a p_x function would be characterized by the integers (1,0,0), for example. The first subscript in φ_{ak} will be used to symbolize **A** and **a**, while the

 0021-9606/88/215777-10$02.10

second refers to the exponent α_k, and will be dropped when the index is not explicitly of interest.

The contracted basis functions ϕ_a used in molecular calculations[17] are fixed linear combinations of primitive Gaussians φ_{ak}, with all primitives having the same angular momentum indices **a**, and the same center **A**, but different exponents α_k:

$$\phi_a(\mathbf{r}) = \sum_k^K D_{ak}\varphi_{ak}(\mathbf{r}). \tag{3}$$

The length of the linear combination K is called the degree of contraction, and the D_{ak} are the contraction coefficients. The use of contracted functions is primarily to allow the basis functions to resemble atomic orbitals. "Shells" of basis functions have common exponents, and thus a p shell consists of a p_x, p_y, and a p_z function, each having identical primitive exponents and a common center. In this case, the contraction coefficients must also be the same for each p function to ensure an isotropic basis. Finally, for computational efficiency, sp shells are often defined, in which an s function shares exponents and centers with a p shell, but can be contracted differently.

A primitive ERI over four primitive unnormalized Gaussians is a six-dimensional integral, which for brevity will be denoted by $[\mathbf{a}_k\mathbf{b}_l|\mathbf{c}_m\mathbf{d}_n]$:

$$[\mathbf{a}_k\mathbf{b}_l|\mathbf{c}_m\mathbf{d}_n] = \int\int \varphi_{ak}(\mathbf{r}_1)\varphi_{bl}(\mathbf{r}_1) \times r_{12}^{-1}\varphi_{cm}(\mathbf{r}_2)\varphi_{dn}(\mathbf{r}_2)d\mathbf{r}_1 d\mathbf{r}_2. \tag{4}$$

If the subscripts are not of interest, they will be suppressed, leaving the notation for a primitive ERI as $[\mathbf{ab}|\mathbf{cd}]$. A contracted ERI, $(\mathbf{ab}|\mathbf{cd})$, will be distinguished from primitive ERIs by using round instead of square brackets. From Eq. (3), the connection between them is

$$(\mathbf{ab}|\mathbf{cd}) = \sum_k^K \sum_l^L \sum_m^M \sum_n^N D_{ak}D_{bl}D_{cm}D_{dn}[\mathbf{a}_k\mathbf{b}_l|\mathbf{c}_m\mathbf{d}_n]. \tag{5}$$

We shall not rederive the RR for ERIs obtained by OS in Ref. 9, but rather just present the key result, which is closely related to Schlegel's derivative theory.[10,11] It is

$$\begin{aligned}[(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]^{(m)} &= (P_i - A_i)[\mathbf{ab}|\mathbf{cd}]^{(m)} + (W_i - P_i)[\mathbf{ab}|\mathbf{cd}]^{(m+1)} \\ &+ \frac{a_i}{2\zeta}\left([(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]^{(m)} - \frac{\eta}{\zeta+\eta}[(\mathbf{a}-\mathbf{l}_i)\mathbf{b}|\mathbf{cd}]^{(m+1)}\right) \\ &+ \frac{b_i}{2\zeta}\left([\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}]^{(m)} - \frac{\eta}{\zeta+\eta}[\mathbf{a}(\mathbf{b}-\mathbf{l}_i)|\mathbf{cd}]^{(m+1)}\right) \\ &+ \frac{c_i}{2(\zeta+\eta)}[\mathbf{ab}|(\mathbf{c}-\mathbf{1}_i)\mathbf{d}]^{(m+1)} + \frac{d_i}{2(\zeta+\eta)}[\mathbf{ab}|\mathbf{c}(\mathbf{d}-\mathbf{1}_i)]^{(m+1)},\end{aligned} \tag{6}$$

where i is one of x, y, z and

$$\mathbf{1}_i = (\delta_{ix},\delta_{iy},\delta_{iz}), \tag{7}$$

$$\zeta = \alpha + \beta, \tag{8}$$

$$P_i = \frac{\alpha A_i + \beta B_i}{\alpha+\beta}. \tag{9}$$

η and Q_i are the analogs of ζ and P_i, constructed from φ_c and φ_d instead of φ_a and φ_b. Finally,

$$W_i = \frac{\zeta P_i + \eta Q_i}{\zeta+\eta}. \tag{10}$$

The meaning of the superscript index m is that integrals with $m=0$ are true ERIs of the form (4), while integrals with $m>0$ are the auxiliary integrals defined by OS. These are necessary to obtain true ERIs via Eq. (6). The recurrence relation (6) expresses primitive ERIs of higher angular momentum as a linear combination of integrals of lower angular momentum, and for this reason we shall refer to it as the *vertical* recurrence relation (VRR).

It should be noted that the constants involved in the VRR, and defined in Eqs. (7)–(10), are all independent of the angular momenta of the primitive Gaussians. Therefore, the number of such constants does not increase with the total angular momentum of the ERI:

$$L = \sum_i^3 (a_i + b_i + c_i + d_i) \tag{11}$$

as pointed out by OS. The other quantities necessary to apply the VRR, are the s-type integrals

$$[ss|ss]^{(m)} = (\zeta+\eta)^{-1/2}K_{AB}K_{CD}F_m(T), \tag{12}$$

where

$$T = \frac{\zeta\eta}{\zeta+\eta}(\mathbf{P}-\mathbf{Q})^2, \tag{13}$$

$$F_m(T) = \int_0^1 t^{2m}\exp(-Tt^2)dt, \tag{14}$$

$$K_{AB} = 2^{1/2}\frac{\pi^{5/4}}{\alpha+\beta}\exp\left[-\frac{\alpha\beta}{\alpha+\beta}(\mathbf{A}-\mathbf{B})^2\right], \tag{15}$$

and K_{CD} is defined similarly. There are many efficient methods for evaluating the $F_m(T)$,[5,6,9] and we use an approach similar to that described by OS.[9]

It is possible to obtain another RR from Eq. (6) by subtracting the VRR for $[\mathbf{a}(\mathbf{b}+\mathbf{1}_i)|\mathbf{cd}]^{(m)}$:

$$[\mathbf{a}(\mathbf{b}+\mathbf{1}_i)|\mathbf{cd}]^{(m)} = (P_i - B_i)[\mathbf{ab}|\mathbf{cd}]^{(m)} + (W_i - P_i)[\mathbf{ab}|\mathbf{cd}]^{(m+1)} + \frac{a_i}{2\zeta}\left([(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]^{(m)} - \frac{\eta}{\zeta+\eta}[(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]^{(m+1)}\right) + \frac{b_i}{2\zeta}\left([\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}]^{(m)} - \frac{\eta}{\zeta+\eta}[a(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}]^{(m+1)}\right) + \frac{c_i}{2(\zeta+\eta)}[\mathbf{ab}|(\mathbf{c}-\mathbf{1}_i)\mathbf{d}]^{(m+1)} + \frac{d_i}{2(\zeta+\eta)}[\mathbf{ab}|\mathbf{c}(\mathbf{d}-\mathbf{1}_i)]^{(m+1)} \quad (16)$$

from that for $[(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]^{(m)}$, Eq. (6), which is in terms of the same integrals. This yields

$$[\mathbf{a}(\mathbf{b}+\mathbf{1}_i)|\mathbf{cd}]^{(m)} = [(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]^{(m)} + (A_i - B_i)[\mathbf{ab}|\mathbf{cd}]^{(m)} \quad (17)$$

which is a new recurrence relation expressing a given integral in terms of another integral of the same total angular momentum, but with a unit of angular momentum shifted from the first to the second position, plus a second integral of lower angular momentum. We shall therefore call Eq. (17) the *horizontal* recurrence relation (HRR), to stress its use in shifting angular momentum from position 1 to 2 (or 3 to 4).

An important feature of the HRR [Eq. (17)] is that the constant in the second term involves only the basis function centers. As a result, the HRR can be applied to contracted integrals and written as

$$(\mathbf{a}(\mathbf{b}+\mathbf{1}_i)|\mathbf{cd}) = [(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}) + (A_i - B_i)(\mathbf{ab}|\mathbf{cd}). \quad (18)$$

In this equation, the *same* contraction coefficients must be used for the pairs $(\mathbf{a},\mathbf{a}+\mathbf{1}_i)$ and $(\mathbf{b},\mathbf{b}+\mathbf{1}_i)$ and for $\mathbf{c}$ and $\mathbf{d}$ on both sides [$\mathbf{a}$ is used here as shorthand for the contracted function ϕ_a, defined by Eq. (3)].

III. ALGORITHM FOR ERI EVALUATION USING RECURRENCE RELATIONS

The problem at this stage is to develop an algorithm that uses the VRR and the HRR to obtain any desired contracted ERI with a minimum number of floating point operations. Actually, since basis functions in the same shell have much in common, it will be more efficient to produce all ERIs associated with a given shell quartet at once. If the four shells all contain only functions of a single angular momentum, we shall refer to the set of target ERIs as a *class*. This class will be denoted as $(ab|cd)$ where a,b,c,d will now refer to the angular momentum of each shell. For example, four p shells will give rise to the class $(11|11)$, or $(pp|pp)$. If any of the shells involve multiple angular momenta (such as sp shells), then the shell quartet will lead to a number of classes. For example, the shell quartet (sp,sp,s,s) will require the four classes $(ss|ss)$, $(ps|ss)$, $(sp|ss)$, and $(pp|ss)$, all being formed from a common set of primitives.

Explicitly minimizing the work needed to form a class of ERIs is a complicated tree-search problem, involving the investigation of all possible reduction sequences. We shall not attempt this, but rather seek an efficient general procedure that is simple and fast enough to include as a logic subroutine in an ERI program. Full optimization of the reduction sequence is an interesting problem for future investigation.

In a general algorithm, it is important to exploit the fact that the HRR can be applied to contracted integrals, rather than individual primitives. This use of the HRR will not be costly, since in practical calculations with even moderate degrees of contraction, the effort to form and contract the primitive ERIs will dominate any subsequent manipulations. Perhaps the simplest such use of the HRR, which shifts angular momentum from position 1 to 2, or 3 to 4, would be to convert contracted integrals of the form $((a+b)0|(c+d)0)$ to $(ab|cd)$. The initial problem would then be to form the $((a+b)0|(c+d)0)$ integrals using the VRR.

Does this use of the HRR reduce the work that must be done by the VRR? Yes, since if the angular momentum of positions 2 and 4 is always zero, then the number of terms in the VRR [Eq. (6)] is reduced from a maximum of 8 to a maximum of 5. Additionally and more importantly, far fewer primitive quantities must be calculated than if the VRR were used alone. For example, 36 primitive $[ds|ds]$ will be contracted to 36 $(ds|ds)$ and then converted by the HRR into 81 $(pp|pp)$ ERIs.

Let us now summarize the proposed algorithm for evaluating ERIs for a shell quartet involving only a single class $(ab|cd)$ [e.g., $(pp|pp)$]. Three main steps are required. The order in which they are listed below is the order of execution in the *logic* subroutine, which is beginning from $(ab|cd)$, and working backwards. This makes it straightforward to determine the required intermediate integral classes. For ERI *production* on the other hand, the steps are performed in the reverse order.

HRR step: Relate $(ab|cd)$ via the HRR to classes of the form $(e0|f0)$, by shifting angular momentum from position 2 to 1, and 4 to 3. From inspection of the HRR (18), it is evident that generally there will be $(b+1)(d+1)$ such classes, ranging from $((a+b)0|(c+d)0)$ down to $(a0|c0)$.

Contraction step: The set of contracted classes $(e0|f0)$ must be obtained from the corresponding set of primitive classes $[e0|f0]$ via Eq. (5). The contraction coefficients used are of course those appropriate for the target class $(ab|cd)$.

VRR step: Each primitive class $[e0|f0]$, needed for the contraction step, is reduced to lower classes by the VRR. The resulting lower classes are in turn reduced by the VRR, until no further reduction is possible. This process terminates with the $[ss|ss]^{(m)}$ integrals, for which m will range from 0 to $a+b+c+d$. The result is a list of primitive

classes, each with an associated range of m values, and each (except $[ss|ss]^{(m)}$) being given in terms of other classes of lower angular momentum in the list.

As a simple example of the use of this algorithm, Table I symbolically illustrates the sequence of operations associated with processing $(dd|ss)$ integrals. The sequence is written in the *logic* order, beginning from the desired class $(ab|cd)$ and working back through the HRR, contraction, and VRR steps. This leads to lists of intermediate contracted classes, the classes which are actually contracted, and finally the primitive classes, concluding with $[ss|ss]^{(m)}$. For *production*, the order is reversed, and the $[ss|ss]^{(m)}$ ERIs are first explicitly calculated, which allows all other classes in the primitive list to be constructed in order of increasing angular momentum via the VRR. After contraction, the HRR is applied to eventually yield the target $(ab|cd)$ ERIs.

An issue that has not yet been addressed is the question of whether to reduce index 1 or index 3 of $[e0|f0]$ when applying the VRR. The choice is made on the basis of two tests:

(1) Reduction is at the position which introduces fewest additional integrals at the next lowest angular momentum level.

(2) If (1) is inconclusive, reduction is at the position which involves fewest terms in the VRR.

For example, given a class $[ds|ps]^{(0)}$ to reduce, two possible situations, and the resulting actions, are:

(a) $[ps|ps]^{(0)}$ is already in the list, and $[ds|ps]^{(0)}$ is not. Reduction at position 1 leads to $[ps|ps]$, and is preferred by test (1). The reverse situation would lead to reduction at position 3.

(b) Both lower classes $[ds|ss]^{(0,1)}$ and $[ps|ps]^{(0,1)}$ are already in the list. Test (1) fails, and test (2) leads to reduction at position 3, since three rather than five terms then occur in the VRR. In the case of the HRR, which is outside the contraction loops, and hence not so important, the fourth index is (by arbitrary convention) stepped up before the second.

The generalization of this algorithm to shell quartets involving multiple classes is as follows. The HRR and contraction steps are executed separately for each class using appropriate contraction coefficients. In the process, a full list of required primitive classes $[e0|f0]$ is accumulated, just as it was for a single class. The VRR step is then the reduction of this full set of primitives to the s-type integrals [Eq. (12)].

TABLE I. Generation of $(dd|ss)$ integrals from $[ss|ss]$ integrals. The functional notation VRR( ) and HRR( ) indicates applying the VRR and the HRR with the required lower ERI classes listed as arguments. All contracted classes have $m = 0$.

Target class: $(dd|ss)$
Applying the horizontal recurrence relation: $(dd|ss) = \mathrm{HRR}\{(fp|ss), (dp|ss)\}$ $(fp|ss) = \mathrm{HRR}\{(gs|ss), (fs|ss)\}$ $(dp|ss) = \mathrm{HRR}\{(fs|ss), (ds|ss)\}$
Classes to be formed by contraction of primitive integrals: $(gs|ss)$, $(fs|ss)$, $(ds|ss)$
Applying the vertical recurrence relation: $[gs|ss]^{(0)} = \mathrm{VRR}\{[fs|ss]^{(0,1)}, [ds|ss]^{(0,1)}\}$ $[fs|ss]^{(0,1)} = \mathrm{VRR}\{[ds|ss]^{(0,2)}, [ps|ss]^{(0,2)}\}$ $[ds|ss]^{(0,2)} = \mathrm{VRR}\{[ps|ss]^{(0,3)}, [ss|ss]^{(0,3)}\}$ $[ps|ss]^{(0,3)} = \mathrm{VRR}\{[ss|ss]^{(0,4)}\}$ $[ss|ss]^{(0,4)}$ (to be formed directly)

IV. EXTENSION TO DERIVATIVE EVALUATION

Since the derivative of a primitive Gaussian φ_a [written below as $(\mathbf{a})$] is a linear combination of a higher and a lower Gaussian:

$$\frac{\partial}{\partial A_i}(\mathbf{a}) = 2\alpha(\mathbf{a}+\mathbf{1}_i) - a_i(\mathbf{a}-\mathbf{1}_i)\,, \tag{19}$$

it follows that the first derivative of a primitive ERI is also a linear combination of higher and lower ERIs:

$$\frac{\partial}{\partial A_i}[\mathbf{ab}|\mathbf{cd}] = 2\alpha[(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}] - a_i[(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}]. \tag{20}$$

Accordingly, the algorithm outlined above can be used to generate first derivatives, if the necessary additional target integrals are generated. In the context of the HF method, the derivatives will be summed directly into Cartesian forces.[2]

The number of additional target classes can be reduced by translational invariance[18]:

$$\frac{\partial}{\partial A_i}(\mathbf{ab}|\mathbf{cd}) + \frac{\partial}{\partial B_i}(\mathbf{ab}|\mathbf{cd}) + \frac{\partial}{\partial C_i}(\mathbf{ab}|\mathbf{cd}) + \frac{\partial}{\partial D_i}(\mathbf{ab}|\mathbf{cd}) = 0. \tag{21}$$

In the worst case of an ERI with each of the four basic functions on a different center, only three rather than four centers must be differentiated, with the derivative at the fourth center being the negative sum of the other three. If two or more of the centers are common, then Eq. (21) will involve fewer terms, and one just has to differentiate with respect to the two or fewer unique centers. So, at most six additional target classes will be necessary to generate the derivatives of a given class. There are three extra classes where the angular momentum of each center being differentiated has been incremented, and three where it has been decremented (if this is possible).

Extension of the algorithm to allow derivative evaluation also requires a modification of the contraction step, since the higher integral in the first term of Eq. (20) is scaled by twice the primitive exponent α. Thus, all primitive classes that are necessary to yield the higher class via the HRR must be contracted using contraction coefficients scaled by the appropriate primitive exponent. The HRR is then applied, taking care to use appropriately contracted input classes. Finally, in an additional fourth step, the derivatives of contracted ERIs are formed by the contracted analog of Eq. (20):

$$\frac{\partial}{\partial A_i}(\mathbf{ab}|\mathbf{cd}) = ((\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd})_\alpha - a_i((\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}), \tag{22}$$

where the subscript α on the higher integral class indicates

that it has been formed with contraction coefficients scaled by twice the primitive exponents of the functions on center **A**.

Since our recently developed method for simultaneous optimization of HF wave function and geometry[19] requires both ERIs and their first derivatives on each cycle, we shall focus on computing them together, rather than separately which can be easily done also. As an example, Table II shows the classes of integrals necessary to evaluate both integrals and derivatives of $(pp|ss)$ ERIs, for the case of four distinct centers.

In addition to the substantial savings arising due to translational invariance, further economies are possible when there are common centers. The HRR [Eq. (18)] reduces to an equality if **A** and **B** are coincident centers, and similarly the first term of the VRR [Eq. (6)] is zero. As integrals involving four distinct centers do not predominate until the molecular size is quite large (typically over 11 atoms[20]), it is very worthwhile to exploit these simplifications.

In the special case of four uncontracted shells, the work associated with processing the HRR is significant, and the algorithm outlined thus far is not optimal. It can be significantly improved, particularly for derivative evaluation and when constrained *sp* shells are present, if the HRR is applied before contraction rather than after, due to the fact that fewer intermediate classes arise in the HRR stage. In the context of Table II, obtaining derivatives of $(pp|ss)$ ERIs requires the formation of $(dp|ss)_\alpha$ and $(pd|ss)_\beta$ ERI classes, which are each generated from the same primitive quantities, but using different sets of contraction coefficients. Contracting after applying the HRR therefore allows us to exploit this commonality and reduce the total work involved.

TABLE II. Generation of $(pp|ss)$ ERIs and first derivatives. ERIs with subscript α, β, and γ are formed with exponent-scaled contraction coefficients from centers 1, 2, and 3, respectively.

Target classes: $(pp|ss)$, $(dp|ss)_\alpha$, $(sp|ss)$,
$(pd|ss)_\beta$, $(ps|ss)$, $(pp|ps)_\gamma$

Applying the horizontal recurrence relation:
$(pd|ss)_\beta = \mathrm{HRR}\{(dp|ss)_\beta, (pp|ss)_\beta\}$
$(dp|ss)_\alpha = \mathrm{HRR}\{(fs|ss)_\alpha, (ds|ss)_\alpha\}$
$(dp|ss)_\beta = \mathrm{HRR}\{(fs|ss)_\beta, (ds|ss)_\beta\}$
$(pp|ps)_\gamma = \mathrm{HRR}\{(ds|ps)_\gamma, (ps|ps)_\gamma\}$
$(pp|ss) = \mathrm{HRR}\{(ds|ss), (ps|ss)\}$
$(pp|ss)_\beta = \mathrm{HRR}\{(ds|ss)_\beta, (ps|ss)_\beta\}$
$(sp|ss) = \mathrm{HRR}\{(ps|ss), (ss|ss)\}$

Classes to be formed by contraction of primitive integrals:
$(fs|ss)_\alpha$, $(fs|ss)_\beta$, $(ds|ps)_\gamma$, $(ds|ss)$, $(ds|ss)_\alpha$,
$(ds|ss)_\beta$, $(ps|ps)_\gamma$, $(ps|ss)$, $(ps|ss)_\beta$, $(ss|ss)$

Applying the vertical recurrence relation:
$[fs|ss]^{(0)} = \mathrm{VRR}\{[ds|ss]^{(0,1)}, [ps|ss]^{(0,1)}\}$
$[ds|ps]^{(0)} = \mathrm{VRR}\{[ds|ss]^{(0,1)}, [ps|ss]^{(1)}\}$
$[ds|ss]^{(0,1)} = \mathrm{VRR}\{[ps|ss]^{(0,2)}, [ss|ss]^{(0,2)}\}$
$[ps|ps]^{(0)} = \mathrm{VRR}\{[ps|ss]^{(0,1)}, [ss|ss]^{(1)}\}$
$[ps|ss]^{(0,2)} = \mathrm{VRR}\{[ss|ss]^{(0,3)}\}$
$[ss|ss]^{(0,3)}$ (to be formed directly)

V. PERFORMANCE ASSESSMENT: ERI EVALUATION

A floating point operation (FLOP) count is simply the sum of the number of floating point adds, subtracts, multiplies, and divides required for a particular problem. FLOP counts are a useful *theoretical* measure of the efficiency of ERI algorithms, since the results are somewhat independent of the details of implementation. Then, to assess the efficiency of *implementation*, the amount of computer time (CPU time) must be examined.

FLOP counts are available in the literature for several ERI algorithms, with the work of Hegarty and van der Velde[8] being particularly valuable in this regard. They reported FLOP counts for the generation of $(ss|ss)$, $(pp|pp)$, $(dd|dd)$, and $(ff|ff)$ integral classes by three separate methods. The two most efficient were the Saunders algorithm[7] (a variant of the Rys approach[14]) and the McMurchie–Davidson (MD) method,[21] which they implemented. The FLOP counts are expressed as

$$N = xK^4 + yK^2 + z, \tag{23}$$

where each of the four shells is assumed to have degree of contraction K.

Table III collects the coefficients x, y, z for the Saunders and MD methods, and our algorithm (henceforth referred to as HGP), for the $(pp|pp)$, $(sp,sp|sp,sp)$ [abbreviated $(sp)^4$], $(dd|dd)$, and $(ff|ff)$ cases. The trivial $(ss|ss)$ case is not shown. For $(pp|pp)$ and $(sp)^4$, the PH method[12] is also included, although it treats *p* shells as *sp*, and therefore calculates extra integrals in the $(pp|pp)$ case. The FLOP counts assume that each of the four shells is on a distinct center. Significant reductions occur in the HGP (see Sec. IV) and MD methods if centers are common, but not in the Saunders method.

If the degree of contraction K is large, we can focus just on the K^4 coefficient in Table III in comparing the different methods. For $(pp|pp)$ and $(sp)^4$ the PH method is then the most efficient, which is expected since this is the type of

TABLE III. Floating point operation counts for ERI formation.[a,b,c] The coefficients x, y, and z refer to Eq. (23) of the text.

Class	Coeff	PH	Saunders	MD	HGP
$(pp\|pp)$	$x\ (\times K^4)$	220	1 800	1 100	920
	$y\ (\times K^2)$	2300	50	600	30
	$z\ (\times K^0)$	4000	0	0	330
$(sp)^4$	$x\ (\times K^4)$	220	3 600	1 500	1 400
	$y\ (\times K^2)$	2300	50	1 700	30
	$z\ (\times K^0)$	4000	0	0	800
$(dd\|dd)$	$x\ (\times K^4)$	···	30 900	27 300	14 600
	$y\ (\times K^2)$	···	220	24 000	30
	$z\ (\times K^0)$	···	0	0	11 300
$(ff\|ff)$	$x\ (\times K^4)$	···	276 000	342 000	108 000
	$y\ (\times K^2)$	···	600	383 000	30
	$z\ (\times K^0)$	···	0	0	135 000

[a] The MD and Saunders results are from Ref. 8(a), except for the $(sp)^4$ case, which we treated using the loop structures of Ref. 8.

[b] The floating point operation counts include one multiply and one addition for contracting each primitive integral. In Ref. 8(a), negative z coefficients appear because it is not necessary to do this with the *first* set of primitives.

[c] Three significant figures are generally given. Smaller numbers, such as those for $(pp|pp)$, do not have this precision, however.

problem for which it was designed.[12] The HGP method is slightly superior to the MD method, and significantly better than the Saunders approach. For $(dd|dd)$ and $(ff|ff)$, the HGP method is far superior to either the Saunders or the MD method, reflecting the benefits of applying the HRR outside the contraction loops.

If the degree of contraction is low, then the work proportional to K^2 and 1 (y and z coefficients) becomes important also. For all three examples, the HGP method requires fewer FLOPS than either the MD or Saunders methods, for *any* degree of contraction. The Rys-based approaches such as the Saunders method, had been fastest for ERIs involving d or f functions, until the work of OS.[9] These results suggest the HGP method will be considerably better than the Rys family of methods, at least up to the level of f functions. For s and p functions, the HGP method will be competitive with the PH method for low degrees of contraction (K between 1 and 2), but not for higher K. However, the basis of the PH method (use of special axes) is also applicable to our method, and we are currently investigating this idea.

One very important question which has not yet been addressed, is how the HGP method compares with the original OS method.[9] As OS did not report FLOP counts, and did not fully specify their algorithm, we can only make a rough comparison. The difference between the OS and HGP approaches is that we use the HRR outside the contraction loops, to reduce the K^4 work (VRR and contraction steps). Therefore there will be little difference for $(pp|pp)$ and $(sp)^4$, since most of the work in our algorithm is still in the VRR and contraction steps. For $(dd|dd)$ and $(ff|ff)$, the HRR step becomes increasingly important (reflected in Table III by the proportionately larger sizes of the z coefficients), and significant savings should be achieved. We estimate the HGP method to require about 25% fewer FLOPS for uncontracted $(dd|dd)$, and 35% fewer FLOPS for $(ff|ff)$, relative to the use of the VRR alone. For contracted shell quartets, the difference is of course much greater.

Some details of the computer implementation of the HGP method are given in the Appendix. Our program can calculate the ERIs and return a Fock matrix,[3] or compute first derivatives of ERIs to yield Cartesian forces,[2] or do both at once.[19] It can currently handle s, p, and d functions; extension to f functions is readily possible. Versions are running on Vax and Alliant computers, at present. For timing purposes, we shall report results from the MicroVax II, a scalar computer, and the Alliant FX-8, which consists of up to eight loosely coupled vector processors (vector length 32), sharing common main memory. The FX-8 timings given below are based on the use of a single processor, although we are currently developing code suitable for multiple processors. Within GAUSSIAN 86, scalar versions of the PH and Rys methods are available, as well as a vector-oriented version[22] of the Rys method for direct SCF. The latter can be compared directly with our program; comparison with the first two codes is complicated by the fact that they do not form a Fock matrix. However, this is significant only when the degree of contraction is low.

The first set of timings in Table IV are for four jobs involving just $(ss|ss)$, $(pp|pp)$, $(sp)^4$, and $(dd|dd)$ ERIs,

TABLE IV. Timings for ERI evaluation in CPU seconds. The bracketed numbers are normalized relative to the HGP method. See the text for a precise description of the calculations.

ERI	K	Machine	PH[a]	s-Rys[b]	v-Rys[c]	HGP
		μVax II	328	1296	1137	377
			(0.9)	(3.4)	(3.0)	(1.0)
$(ss\|ss)$	4					
		FX-8	52	175	78	29
			(1.8)	(6.1)	(2.7)	(1.0)
		μVax II	419	989	2097	604
			(0.7)	(1.6)	(3.5)	(1.0)
$(pp\|pp)$	2					
		FX-8	81	141	114	36
			(2.2)	(3.9)	(3.2)	(1.0)
		μVax II	494	2047	3492	895
			(0.6)	(2.3)	(3.9)	(1.0)
$(sp)^4$	2					
		FX-8	95	256	214	60
			(1.6)	(4.3)	(3.6)	(1.0)
		μVax II	···	1778	2891	1515
			···	(1.2)	(1.9)	(1.0)
$(dd\|dd)$	1					
		FX-8	···	264	361	81
			···	(3.3)	(4.5)	(1.0)

[a] The Pople–Hehre method; Link 311 of GAUSSIAN 86.
[b] A scalar version of the Rys method: Link 314 of GAUSSIAN 86.
[c] A vector version of the Rys method.

respectively, to allow some comparison with the FLOP counts discussed above. Each case consists of 12 atoms in a bicubic arrangement, with a single shell of basis functions on each atom, using a density matrix constructed by diagonalizing the core Hamiltonian. In the $(ss|ss)$ case, the atoms are hydrogens, with an STO-4G 1s shell[23] ($K = 4$) on each, and an edge length of 0.8 Å. For $(pp|pp)$ [and $(sp)^4$], the atoms are carbons, with an edge length of 1.4 Å. On each carbon is a $2p$ [$2sp$ for $(sp)^4$] shell with the exponents and contraction coefficients taken from the STO-2G basis ($K = 2$). The $(dd|dd)$ geometry is identical to $(pp|pp)$, with a single uncontracted d shell ($K = 1$) of exponent 0.8 on each atom.

The timings of Table IV are generally consistent with the FLOP counts presented above, but naturally show a strong dependence on implementation, with the vector codes performing much better on the FX-8. For the $(ss|ss)$ case, the PH and HGP methods require almost identical numbers of FLOPS, the timing differences reflecting the more vector oriented code of the HGP method. The PH method is slightly faster than the HGP method on the scalar MicroVax for $(pp|pp)$, reflecting its efficient treatment of contracted shells. This edge increases a little for $(sp)^4$, since the PH method is more efficient for sp shells. However, these results are reversed on the FX-8. Relative to both implementations of the Rys method, the HGP method is significantly superior for all three examples [note in the $(dd|dd)$ case that Fock matrix formation is over 30% of the HGP time, which should be deducted when comparing with the scalar Rys code] as expected on the basis of FLOP counts.

Table V is a comparison of the various methods of ERI evaluation for calculations on the *trans* two-pentenal mole-

TABLE V. Timings in CPU seconds, for ERI evaluation on the two-pentenal molecule. Bracketed numbers are normalized relative to the HGP method.

Basis	Machine	PH[a,b]	s-Rys[c]	v-Rys[d]	HGP
STO-3G	μVax II	897	5 635	9 561	1 823
		(0.5)	(3.6)	(5.2)	(1.0)
	FX-8	134	773	546	157
		(0.9)	(4.9)	(3.5)	(1.0)
3-21G	μVax II	2 428	6 413	8 775	2 157
		(1.1)	(3.0)	(4.1)	(1.0)
	FX-8	437	992	697	192
		(2.3)	(5.2)	(3.6)	(1.0)
6-31G*	μVax II	29 245	45 004	83 763	14 963
		(2.0)	(3.0)	(5.6)	(1.0)
	FX-8	4 167	6 561	6 009	1 148
		(3.6)	(5.7)	(5.2)	(1.0)

[a] The Pople–Hehre method; Link 311 of GAUSSIAN 86.
[b] The PH timing for 6-31G* uses the PH method for ERIs containing just s and p functions, and the scalar Rys method for ERIs containing d functions.
[c] A scalar version of the Rys method; Link 314 of GAUSSIAN 86.
[d] A vector version of the Rys method.

cule ($CH_3CH_2CH=CHCHO$, methyl *gauche* and CO *syn* to double bond, HF/6-31G* optimized geometry) using three different basis sets. Two contain just s and p functions of which the first is the minimal STO-3G basis,[23] in which each AO is represented by a single basis function composed of three primitives. The split valence 3-21G basis[24] treats the core similarly but has two basis functions per valence AO: an inner function composed of two primitives, and an uncontracted outer function. The third basis is 6-31G*, which is constructed analogously to the 3-21G basis, but with six primitives per core basis function, and three for the inner valence function. Additionally, a set of six Cartesian d functions is added to elements heavier than helium.[25] All three of these basis sets use sp rather then pure p shells. For the 6-31G* job, the PH entry in Table V corresponds to using the PH method for ERIs which do not involve d functions, and the scalar Rys method for those that do. Again, the PH and scalar Rys times exclude Fock matrix formation, unlike the HGP and vector Rys methods.

On the scalar MicroVax, the PH method is superior to the HGP method for the highly contracted ($K=3$) STO-3G basis. However, for the less contracted 3-21G basis the HGP method is faster despite including Fock matrix formation (25% of the total). On the Alliant, the vectorized HGP implementation gives it a further advantage. The two Rys codes are not very competitive. The HGP method is several times faster than either code, on either machine. It is also two and over three times as fast as the scalar PH/Rys combination (which is the best GAUSSIAN 86 d function method) for the 6-31* job, on the MicroVax and FX-8, respectively. Unlike OS,[9] we do not observe lower speedups relative to the Rys method for basis sets containing d functions. This may indirectly confirm that the HGP method is more efficient than the OS approach for ERIs over d functions.

VI. PERFORMANCE ASSESSMENT: DERIVATIVE EVALUATION

Relatively few FLOP counts for derivative evaluation are available in the literature. Results are available[26] for the method of Schlegel,[11] which applies to s and p functions. We have also made some estimates for a Rys-polynomial method.[16] These are collected in Table VI, along with results for the HGP method for first derivatives of $(pp|pp)$, $(sp)^4$, $(dd|dd)$, and $(ff|ff)$, in the case that all four centers are distinct. Like the PH method for ERIs, the Schlegel method treats all p shells as sp ones, and calculates more derivatives than necessary in the $(pp|pp)$ example.

In these examples, the HGP method requires fewer FLOPs than the Rys method for *any* degree of contraction, K. As much of the HGP method's work is outside the contraction loops (z coefficient), it will gain extra advantage as K rises. Although the HGP method involves far fewer FLOPs than the Rys method for $(ff|ff)$, so much storage (10^5 words per shell quartet) is required that our approach may not be practical beyond $(dd|dd)$ derivatives. The HGP method is superior to Schlegel's for $(pp|pp)$ derivatives, although this is not a favorable case for the latter. They are similar for derivatives of $(sp)^4$, for which Schlegels' method was optimized. Relative to the derivative method of OS,[9] it is likely that the use of the HRR gives the HGP method a slight advantage for sp basis sets, which increases on going to d functions. The OS and Schlegel methods should be similar[10] for $(sp)^4$ and $(pp|pp)$.

Tables VII and VIII are the analogs of Tables IV and V, for the calculation of first derivatives of ERIs, and associated summation into HF Cartesian forces. The Schlegel method for s and p functions is combined with the scalar Rys code for d functions in the 6-31G* job of Table VIII. The results are generally similar to those for ERI evaluation, with the HGP method tending to be superior to the Rys method for s, p, and d functions, by factors of 2 to 5 generally. The HGP and Schlegel methods appear comparable for s and p functions,

TABLE VI. Floating point operation counts for derivative evaluation.[a] The coefficients x, y, and z refer to Eq. (23).

Class	Coeff	Schlegel	Rys[b]	HGP[c]
$(pp\|pp)$	$x\ (\times K^4)$	8030	10 200	3 440
	$y\ (\times K^2)$	120	80	30
	$z\ (\times K^0)$	0	0	4 600
$(sp)^4$	$x\ (\times K^4)$	8030	24 900	6 600
	$y\ (\times K^2)$	120	80	30
	$z\ (\times K^0)$	0	0	13 100
$(dd\|dd)$	$x\ (\times K^4)$	···	203 000	45 000
	$y\ (\times K^2)$	···	140	30
	$z\ (\times K^0)$	···	0	109 000
$(ff\|ff)$	$x\ (\times K^4)$	···	1 850 000	284 000
	$y\ (\times K^2)$	···	210	30
	$z\ (\times K^0)$	···	0	1 100 000

[a] The totals assume all four centers are distinct, and include summation of the derivatives into Cartesian forces, but not two particle density matrix formation.
[b] The Rys method of Ref. 16.
[c] If the shells are uncontracted, the methods of Sec. IV can be applied to save about 600, 6300, 26 000, and 220 000 operations in the $(pp|pp)$, $(sp)^4$, $(dd|dd)$, and $(ff|ff)$ cases, respectively.

TABLE VII. Timings in CPU seconds for first derivative evaluation. The four systems are the same as used previously in Table IV.

ERI	K	Machine	S[a]	s-Rys[b]	v-Rys[c]	HGP
		μVax II	617	2486	1939	1037
			(0.6)	(2.4)	(1.9)	(1.0)
$(ss\|ss)$	4					
		FX-8	77	402	150	66
			(1.2)	(6.1)	(2.3)	(1.0)
		μVax II	2287	3342	7 047	2033
			(1.2)	(1.6)	(3.5)	(1.0)
$(pp\|pp)$	2					
		FX-8	484	435	348	115
			(4.2)	(3.8)	(3.0)	(1.0)
		μVax II	2316	7500	16 128	3440
			(0.7)	(2.2)	(4.7)	(1.0)
$(sp)^4$	2					
		FX-8	487	930	765	237
			(2.1)	(3.9)	(3.2)	(1.0)
		μVax II	...	4739	9 393	5393
			...	(0.9)	(1.7)	(1.0)
$(dd\|dd)$	1					
		FX-8	...	568	433	263
			...	(2.2)	(1.7)	(1.0)

[a] Schlegel's method; Link 702 of GAUSSIAN 86.
[b] A scalar version of the Rys method; Link 703 of GAUSSIAN 86.
[c] A vector version of the Rys method; also part of Link 703.

although our vector-oriented code is significantly slower on the scalar MicroVax, for the examples of Table VIII.

VII. CONCLUSIONS

Based on the Obara–Saika[9] recurrence relation, and a new recurrence relation derived from it, we have formulated and implemented an efficient algorithm (abbreviated HGP) for two-electron integral and integral derivative evaluation. It has the following features:

(1) For basis sets involving d functions or higher, the HGP method should be significantly faster than the Obara–Saika approach. There will be little difference for sp basis sets.

(2) For ERI calculations, the HGP method is more efficient than the Pople–Hehre method[12] for sp basis sets having low degrees of contraction (e.g., 3-21G), but less efficient for highly contracted basis sets (e.g., STO-3G). As a future refinement, the axis switch technique used in the Pople–Hehre method may be incorporated into the HGP method.

(3) The HGP method is uniformly superior to the Rys method for ERIs[7,14] and derivatives,[16] for sp, and spd basis sets. This has been demonstrated by both floating point operation counts, and program execution times. The operation counts indicate this advantage will be retained for f functions as well.

(4) For basis sets containing d functions, the HGP method is also clearly faster than the combination of the Rys method with special purpose sp methods (Pople–Hehre for ERIs, Schlegel[11] for derivatives).

TABLE VIII. Timings (in CPU seconds) for first derivative calculations on the two-pentenal molecule.

Basis	Machine	S[a,b]	s-Rys[c]	v-Rys[d]	HGP
	μVax II	3 099	15 110	36 735	6 230
		(0.5)	(2.4)	(5.9)	(1.0)
STO-3G					
	FX-8	423	2 160	2 243	451
		(0.9)	(4.8)	(5.0)	(1.0)
	μVax II	4 078	15 537	36 078	6 405
		(0.6)	(2.4)	(5.6)	(1.0)
3-21G					
	FX-8	636	2 351	2 204	436
		(1.5)	(5.4)	(5.1)	(1.0)
	μVax II	84 314	125 005	358 240	45 727
		(1.8)	(2.7)	(7.8)	(1.0)
6-31G*					
	FX-8	11 525	18 032	17 839	3 114
		(3.7)	(5.8)	(5.7)	(1.0)

[a] The Schlegel method; Link 702 of GAUSSIAN 86.
[b] The Schlegel timing for 6-31G* uses the Schlegel method for ERIs containing just s and p functions, and the scalar Rys method for ERIs containing d functions.
[c] A scalar version of the Rys method; Link 703 of GAUSSIAN 86.
[d] A vector version of the Rys method; also part of Link 703.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. CHE-84-09405. We thank Jim Foresman for assistance with the Alliant version of the program.

APPENDIX: COMPUTER IMPLEMENTATION

The general objective of our implementation is to produce a computer program capable of exhibiting good performance on a variety of computer architectures. We shall try to achieve this goal by isolating the compute-intensive parts of the algorithm in several comparatively small and simple subroutines, with the hope that both vector and scalar computers should be able to handle such a code reasonably efficiently.

The fundamental loop structures in an ERI program are over quartets of shells. The work done inside these loops scales up as N^4, and the problem is to minimize this work. Fortunately, many of the quantities required in the N^4 loops can be precomputed in loops over pairs of shells, which requires a relatively insignificant $O(N^2)$ effort. Examples of precomputable quantities are Eqs. (8), (9), and (15), as well as products of contraction coefficients for shell pairs. Storing quantities of size $O(N^2)$ is a feasible proposition, and therefore the program begins with an initial loop over shell pairs to calculate these primitive quantities, and discard any primitive shell pairs for which K_{AB}, Eq. (15), which relates directly to the magnitude of the ERIs through Eq. (12), is sufficiently small. The default integral accuracy is 10^{-10} for compatibility with existing GAUSSIAN 86 standards, with 10^{-14} and 10^{-6} being available as options.

The organization of the N^4 loops is detailed in Table IX, to which the following discussion refers. Table IX omits

TABLE IX. N^4 loop structures in the ERI program. K refers to the degree of contraction of a shell.

```
Do over angular momentum types (2 loops over pairs)
Do over the possible coincident centre cases
    Call Brains (fills indexing arrays)
    Call BiteSz (dimensions scratch arrays)
    Do over the different possible degrees of contraction
        Do until all shell quartets of the present type are done
            Call Chunk (get shell quartet batch of current type)
            Do over current degree of contraction (K^4 loop)
                Call MakeC (build constants needed to apply RR's)
                Call SSSSM (explicitly form [ss|ss](m) integrals)
  K^4       Call VtItgl (form [e0|f0](m) integrals using VRR)
 region     Call MakeCC (make contraction coefficients)
            Call Cntrct (contract [e0|f0] into (e0|f0) ERI's)
            end do
            Call HzItgl (form (ab|cd) ERI's by HRR)
  K^0       Call Dgat2e (sums results into Fock matrix, forces)
 region     Call MovFrc (translational invariance for forces)
        end do
    end do
end do
end do
```

loops that deal with shell quartets involving multiple classes, and uncontracted quartets, as these details are not central to the algorithm. The N^4 loops are based on processing a large number of shell quartets of the same *type* (having identical angular momenta, degrees of contraction, and numbers of common centers) at once. For example, all (p, p, p, p) shell quartets having the same degree of contraction, and no common centers, will be grouped together. Identical operations will be performed on each quartet at every stage of ERI generation, and our simple rate-limiting steps will therefore be based on long inner-most loops over the number of quartets in the current batch. These vectorizable inner loops do not appear explicitly in Table IX, because they are contained within the various subroutines listed (except BRAINS and BITESZ).

The outer loops of Table IX ensure the selection of shell quartets of the same type, based on having initially sorted the shell pairs into common groups in the N^2 setup loops. At this stage, the subroutine BRAINS generates the indexing arrays describing the way in which the VRR and the HRR will be used for shell quartets of the current type. Based on the scratch space required per contracted shell quartet, returned from BRAINS, and the amount of memory available, BITESZ ("bite size") determines the maximum number of shell quartets that can be processed at once. The program then enters a do-until loop in which batches of shell quartets of the current type, selected by CHUNK, are processed to yield ERIs, until all quartets of this type have been dealt with. CHUNK can reject shell quartets based on symmetry criteria[27,28] or small estimated contributions to the Fock matrix or forces.

Within the do-until loop are the compute-intensive operations necessary to form the ERIs for the current batch of shell quartets. The first stage involves work proportional to the product of the degrees of contraction, within a loop over the number of contributing primitive shells. Sets of primitive ERIs are produced by MAKEC, SSSSM, and VTITGL ("vertical integrals") using the VRR step of the algorithm. Their contribution to the necessary contracted ERIs is accumulated in CNTRCT, after the four-center contraction coefficients have been constructed in MAKECC ("make contraction coefficients").

The second stage within the do-until loop is the manipulation of contracted quantities to form the desired ERIs, and optionally their derivatives. This is accomplished in HZITGL ("horizontal integrals"), which implements the HRR step. Finally DGST2E ("digest 2E") explicitly forms the derivatives by Eq. (22) (if requested), and adds the contributions of the current batch of ERIs and derivatives to the Fock matrix and Cartesian forces, respectively. In the context of a conventional SCF job, the ERIs would instead be written to disk, at this stage.

One key issue associated with processing large numbers of shell combinations simultaneously is the large amount of memory needed to store all ERIs and associated auxiliary ERIs at once. This problem becomes particularly acute as the angular momentum of the shell quartets increases. To reduce this requirement, we have implemented an efficient scheme for reusing scratch space in the HRR step, which was originally most space consuming. Two work space areas, W1 and W2, are defined. The VRR yields sets of primitive ERIs in W1, which are used to build the contracted classes $(e0|f0)$ in W2. The HRR is initially used to construct contracted classes of the form $(e1|f0)$ and $(e0|f1)$ in W1, that depend only on the ERIs in W2. The next angular momentum shift goes back into W2, using the information in W1, and so forth. This reduces the scratch space required per $(dd|dd)$ shell quartet by a factor of over 2.5, from about 9300 to 3700 memory locations. The lengths of W1 and W2 are calculated in BRAINS, by recording the maximum of the lengths required at each angular momentum shift.

[1]W. J. Hehre, L. Radom, P. v. R. Schleyer, and J. A. Pople, *Ab Initio Molecular Orbital Theory* (Wiley, New York, 1986).
[2]P. Pulay, in *Modern Theoretical Chemistry*, edited by H. F. Schaefer III (Plenum, New York, 1977), Vol. 4, p. 153.
[3]J. Almlöf, K. Faegri, Jr., and K. Korsell, J. Comput Chem. **3**, 385 (1982).
[4]J. A. Pople, R. Krishnan, H. B. Schlegel, and J. S. Binkley, Int. J. Quantum Chem. Symp. **13**, 225 (1979).
[5]S. F. Boys, Proc. R. Soc. London Ser. A **200**, 542 (1950).
[6]I. Shavitt, in *Methods in Computational Physics*, edited by B. Alder, S. Fernbach, and M. Rotenberg (Academic, New York, 1963), Vol. 2, p. 1.
[7](a) V. R. Saunders, in *Computational Techniques in Quantum Chemistry and Molecular Physics*, edited by G. H. F. Diercksen, B. T. Sutcliffe, and A. Veillard (Reidel, Dordrecht, 1975), p. 347; (b) V. R. Saunders, in *Methods in Computational Physics*, edited by G. H. F. Diercksen and S. Wilson (Reidel, Dordrecht, 1983), p. 1.
[8](a) D. Hegarty and G. Van Der Velde, Int. J. Quantum Chem. **23**, 1135 (1983); (b) D. Hegarty, in *Advanced Theories and Computational Approaches to the Electronic Structure of Molecules*, edited by C. E. Dykstra (Reidel, Dordrecht, 1984), p. 39.

[9]S. Obara and A. Saika, J. Chem. Phys. **84**, 3963 (1986).
[10]H. B. Schlegel (private communication). Equation (25) of Ref. 11 corresponds term by term to the recurrence relation (39) of Ref. 9 [or Eq. (6) in this work]. However, Schlegel's expression is for the derivative of [**ab**|**cd**], *not* the ERI [(**a** + **1**$_i$)**b**|**cd**]. The two are related by Eq. (20) (this paper), and accordingly, Obara and Saika's derivative method should be similar to Schlegel's.
[11]H. B. Schlegel, J. Chem. Phys. **77**, 3676 (1982).
[12]J. A. Pople and W. J. Hehre, J. Comput. Phys. **27**, 161 (1978).
[13]M. Dupuis, D. Sprangler, and J. J. Wedolski, NRCC QC01 GAMESS. As detailed in Ref. 9, a version of GAMESS which had been ported to the Facom M-382 computer was used.
[14](a) M. Dupuis, J. Rys, and H. F. King, J. Chem. Phys. **65**, 111 (1976); (b) H. F. King and M. Dupuis, J. Comput. Phys. **21**, 44 (1976); (c) J. Rys, M. Dupuis, and H. F. King, J. Comput. Chem. **4**, 154 (1983).
[15]Frisch *et al.*, Carnegie-Mellon Quantum Chemisty Publishing Unit, Carnegie-Mellon University, Pittsburgh, PA 15213.
[16]H. B. Schlegel, J. S. Binkley, and J. A. Pople, J. Chem. Phys. **80**, 1976 (1984).
[17]E. R. Davidson, Chem. Rev. **86**, 681 (1986).
[18]A. Korminicki, K. Ishida, K. Morokuma, R. Ditchfield, and M. Conrad, Chem. Phys. Lett. **45**, 595 (1977).
[19]M. Head-Gordon and J. A. Pople, J. Phys. Chem. **92**, 3063 (1988).
[20]M. A. Vincent and H. F. Schaefer, Theor. Chim. Acta **64**, 21 (1983).
[21]L. E. McMurchie and E. R. Davidson, J. Comp. Phys. **26**, 218 (1978).
[22]M. J. Frisch and J. S. Binkley (unpublished).
[23]W. J. Hehre, R. F. Stewart, and J. A. Pople, J. Chem. Phys. **51**, 2657 (1969).
[24]J. S. Binkley, J. A. Pople, and W. J. Hehre, J. Am. Chem. Soc. **102**, 939 (1980).
[25]P. C. Hariharan and J. A. Pople, Theor. Chim. Acta **28**, 213 (1973).
[26]H. B. Schlegel (private communication).
[27]M. Dupuis and H. F. King, Int. J. Quantum Chem. **11**, 613 (1977).
[28]M. Dupuis and H. F. King, J. Chem. Phys. **68**, 3998 (1978).