A tensor approach to two-electron matrix elements

Terry R. Adams
Department of Chemistry, Massey University, Palmerston North, New Zealand

Ross D. Adamson and Peter M. W. Gill
Department of Chemistry, University of Cambridge, Cambridge CB2 1EW, United Kingdom

(Received 3 June 1996; accepted 27 March 1997)

We present a new algorithm, the COLD PRISM, for computing matrix elements in molecular orbital calculations. Whereas traditional approaches form these from two-electron repulsion integrals (ERIs) which, in turn, are formed from shell-pair data, we introduce several alternative paths that do not involve ERIs as intermediates. Tensor multiplication can be used as the basic arithmetic operation on all of the new PRISM paths and the associativity of tensor multiplication plays a key role. We have implemented our approach in the Q-Chem program. © *1997 American Institute of Physics.* [S0021-9606(97)01125-2]

I. INTRODUCTION

It is now well established that high quality *ab initio* molecular orbital calculations[1] can provide theoretical models of chemistry that, for very small systems, often rival experiment in accuracy. Pople's G2 procedure,[2] for example, has been widely and enthusiastically embraced as a general-purpose, ''black box'' model for the investigation of ground-state equilibrium and transition structures and has been successfully applied to numerous chemical problems.

The chief drawback, however, of these highly accurate methods is the staggeringly rapid growth in their computational cost with system size. This has led our group, and a number of others, to take the established methodologies of quantum chemistry and develop modifications with much more modest computational requirements. A variety of such algorithms has been presented recently[3–7] (and more are under development in our laboratory), but each of these still requires a highly efficient algorithm for computing two-electron repulsion integrals (ERIs),

$$(\mathbf{ab}|\mathbf{cd}) = \int\int \phi_{\mathbf{a}}(\mathbf{r}_1)\phi_{\mathbf{b}}(\mathbf{r}_1)\theta(r_{12})\phi_{\mathbf{c}}(\mathbf{r}_2)\phi_{\mathbf{d}}(\mathbf{r}_2)d\mathbf{r}_1\, d\mathbf{r}_2, \tag{1.1}$$

over contracted Gaussian-type basis functions (CGTF),

$$\phi_{\mathbf{a}}(\mathbf{r}) = (x-A_x)^{a_x}(y-A_y)^{a_y}(z-A_z)^{a_z}\sum_{i=1}^{K_a} D_{ai}e^{-\alpha_i|\mathbf{r}-\mathbf{A}|^2}, \tag{1.2}$$

for, although approaches such as CFMM (Ref. 3), KWIK (Ref. 4), and CASE (Refs. 5–7) treat *long*-range Coulomb effects in various ways, their *short*-range treatments all involve integrals of the form (1.1).

ERIs appear to be natural intermediates in the construction of the Fock matrix,[8] and the development of efficient ERI algorithms in the wake of Boys' 1950 paper[9] has attracted much research interest and has recently been reviewed.[10] In the present work, we propose a new ERI algorithm and, more importantly, we introduce a very general scheme in which ERIs are just one of a number of possible intermediates that may be employed in Fock matrix construction.

All methods for computing two-electron matrix elements involve four steps which can be discussed in terms of the charge elements $\rho_i(\mathbf{r})$ in which the electronic density is expanded.

In the operator step O, the momentumless components of pairs of $\rho_i(\mathbf{r})$ are integrated over the two-electron operator. In the PRISM (Refs. 10 and 11), Pople–Hehre,[12] and Dupuis–Rys–King[13] methods (the first includes McMurchie–Davidson,[14] Obara–Saika,[15] Head–Gordon–Pople,[16] and Ten-no[17] methods as special cases), this yields the $[\mathbf{0}]^{(m)}$ integrals,[18] special-axis integrals,[12] or Rys roots and weights,[19] respectively.

In the momentum step L, recursive linear identities are used to transform zero-momentum quantities into ones with the desired angular momenta. Much of the work in integral technology has focused on the construction[10–17,20–23] and efficient utilization[24–26] of such identities.

In the contraction step C, contributions from primitive components are added together (with scaling, if appropriate) to form fully contracted contributions. Deciding how and when to execute the C step optimally lies at the heart of existing PRISM methods.

In the density step D, contracted quantities are scaled by density matrix elements $P_{\mu\nu}$ and further contracted into accumulators. Occasionally, this is known as the ''digestion'' step. As we will show, choices about how and when to execute the D step for optimal efficiency lead to a generalization of existing PRISM schemes.

Approaches can be classified according to the order in which the four steps are executed: the Dupuis–Rys–King, McMurchie–Davidson, and Obara–Saika algorithms are all $OLCD$; Head-Gordon–Pople straddles $OLCD$ and $OCLD$; Pople–Hehre, Ten-no, and Ishida[23] are $OCLD$; and the PRISM algorithms include both $OLCD$ and $OCLD$ paths. But, as shown symbolically in Fig. 1, four other orderings ($OCDL$, $COLD$, $CODL$, and $CDOL$) are possible, provided that mathematical means can be found to perform

$$\begin{tikzcd}
\text{Primitive Gaussians} \arrow[r, "C_1"] \arrow[d, "O_1"] & \text{Contracted Gaussians} \arrow[d, "O_2"] \arrow[dr, "DOL"] & \\
{[m]} \arrow[r, "C_2"] \arrow[d, "L_1"] & \text{Kernels} \arrow[r, "DL"] \arrow[d, "L_2"] & \text{Matrix Elements} \\
{[ab|cd]} \arrow[r, "C_3"] & (ab|cd) \arrow[ur, "D"'] &
\end{tikzcd}$$

FIG. 1. The COLD PRISM.

each of the required steps. Inspired by its most pronounceable path, we will refer to Fig. 1 as the COLD PRISM.

The O_1, L_1, L_2, C_2, C_3, D steps in the COLD PRISM have been considered before: O_1 was first described by Boys and has since been highly optimized,[18] L_1 can be accomplished using the Dupuis–Rys–King, McMurchie–Davidson, or Obara–Saika schemes; L_2 occurs in PRISM methods but, in Sec. II, we present a more efficient matrix-based scheme than either of these. As far as we know, the C_1, O_2, DL, and DOL steps have not previously been discussed in the literature. We note that, whereas the method of Panas and Almlöf[27] (in which multipole expansions are used to treat long-range Coulomb interactions) may appear at first glance to be a $CDOL$ path, it is not one, for it is based on the multipole moments of molecular fragments, rather than of individual shell pairs. As such, it can yield Coulomb, but not exchange, matrix elements.

Using the Einstein summation convention (which we will adopt throughout this paper), the expressions X_iY_i, $X_{ij}Y_j$, and $X_{ij}Y_{jk}$ represent prototypical BLAS-1, BLAS-2, and BLAS-3 constructs.[28–30] It is generally true that the most efficient algorithms on modern computers are those that are based on BLAS-3 (matrix–matrix) or BLAS-2 (matrix–vector) arithmetic. In contrast, schemes that are based on BLAS-1 (vector–vector) arithmetic, rarely achieve more than a modest fraction of the theoretical peak performance of a workstation or supercomputer. Yet, in the GAUSSIAN 92 program,[31] we implemented our early PRISM algorithms using exclusively BLAS-1. Clearly, this leaves much room for improvement. In our view, the ultimate ERI programs will employ high-level BLAS and we have therefore sought, and implemented in the Q-Chem program,[32] matrix (or tensor) formulations for each of the COLD PRISM steps. In the next few sections, we develop practical formulae to accomplish the L_2, O_2, and DL steps and, in each case, indicate how they may be cast in terms of BLAS-2 and BLAS-3 constructs.

II. NOTATION AND DEFINITIONS

In this paper we adopt the notation system that was employed in a recent Review.[10] The CGTF in (1.2) is defined by its angular momentum vector $\mathbf{a}=(a_x,a_y,a_z)$, its position vector $\mathbf{A}=(A_x,A_y,A_z)$, its degree of contraction K_a, its contraction coefficients D_{ai}, and by its exponents α_i. The angular momentum of (1.2) is $a=(a_x+a_y+a_z)$. We refer to the set of CGTFs with the same center and the same set of exponents as a *contracted shell*, e.g., the set of contracted p-functions $\{p_x,p_y,p_z\}$. Equation (1.1) defines an inner product between two functions

$$(\mathbf{ab}|\equiv\phi_{\mathbf{a}}(\mathbf{r}_1)\phi_{\mathbf{b}}(\mathbf{r}_1), \tag{2.1}$$

$$|\mathbf{cd})\equiv\phi_{\mathbf{c}}(\mathbf{r}_2)\phi_{\mathbf{d}}(\mathbf{r}_2), \tag{2.2}$$

where $(\mathbf{ab}|$ and $|\mathbf{cd})$ are termed ''bra'' and ''ket,'' respectively. These functions can be thought of as two charge distributions and the integral (1.1) describes the interaction between the distributions. The complete set of contracted shells on each center for (2.1) or (2.2) is termed a *contracted shell pair*. The degree of contraction of the bra (2.1) is $K_{\text{bra}}=K_aK_b$ and of the ket (2.2) is $K_{\text{ket}}=K_cK_d$. The complete set of contracted shells on each center in the ERI (1.1) is termed a *contracted shell quartet*. The total angular momentum of (1.1) is $L_{\text{tot}}=(a+b+c+d)$ and the total degree of contraction is $K_{\text{tot}}=K_{\text{bra}}K_{\text{ket}}$.

All integrals associated with a shell quartet are called a *class* of integrals. For example, a $(pp|pp)$ class is the set of 81 $(pp|pp)$ integrals associated with four p-shells. It is to our advantage computationally to compute integrals in classes, rather than individually, because all of the integrals in a class share the same four centers and the same set of exponents, and thus generation of integrals involves many common intermediate quantities.

For later reference in our formulation it is useful to define a scaled-bra function,

$$_{a'b'p'}(\mathbf{ab}|=\sum^{K_{\text{bra}}}\frac{(2\alpha)^{a'}(2\beta)^{b'}}{(2\zeta)^{p'}}[\mathbf{ab}|, \tag{2.3}$$

where the shell pairs $[\mathbf{ab}|$ are scaled by exponent ratios. Square brackets distinguish primitive quantities and parenthesis represent contracted quantities. The exponent $\zeta=\alpha+\beta$ arises from the Gaussian product rule.[8,10] A scaled-ket function is similarly defined.

III. THE L_2 STEP

Substituting the powerful identity from Ref. 10,

$$\mathbf{R}\equiv\frac{2\alpha}{2\zeta}(\mathbf{B}-\mathbf{A})+\frac{2\gamma}{2\eta}(\mathbf{C}-\mathbf{D})+(\mathbf{D}-\mathbf{B}), \tag{3.1}$$

into a recast version [Eq. (97) in Ref. 10] of the Obara–Saika recurrence relation,[15] we obtain

$$
\begin{aligned}
{}_{a'b'p'}(\mathbf{a}+\mathbf{1}_i\mathbf{b}|\mathbf{cd})^{(m)}_{c'd'q'} = {} & (B_i-A_i)\{{}_{a'(b'+1)(p'+1)}(\mathbf{ab}|\mathbf{cd})^{(m)}_{c'd'q'} + {}_{(a'+1)b'(p'+2)}(\mathbf{ab}|\mathbf{cd})^{(m+1)}_{c'd'q'}\} + (C_i-D_i) \\
& \times {}_{a'b'(p'+1)}(\mathbf{ab}|\mathbf{cd})^{(m+1)}_{(c'+1)d'(q'+1)} + (D_i-B_i)\ {}_{a'b'(p'+1)}(\mathbf{ab}|\mathbf{cd})^{(m+1)}_{c'd'q'} \\
& + a_i\{{}_{a'b'(p'+1)}(\mathbf{a}-\mathbf{1}_i\mathbf{b}|\mathbf{cd})^{(m)}_{c'd'q'} - {}_{a'b'(p'+2)}(\mathbf{a}-\mathbf{1}_i\mathbf{b}|\mathbf{cd})^{(m+1)}_{c'd'q'}\} \\
& + b_i\{{}_{a'b'(p'+1)}(\mathbf{ab}-\mathbf{1}_i|\mathbf{cd})^{(m)}_{c'd'q'} - {}_{a'b'(p'+2)}(\mathbf{ab}-\mathbf{1}_i|\mathbf{cd})^{(m+1)}_{c'd'q'}\} \\
& + c_i\ {}_{a'b'(p'+1)}(\mathbf{ab}|\mathbf{c}-\mathbf{1}_i\mathbf{d})^{(m+1)}_{c'd'(q'+1)} + d_i\ {}_{a'b'(p'+1)}(\mathbf{ab}|\mathbf{cd}-\mathbf{1}_i)^{(m+1)}_{c'd'(q'+1)}. \qquad (3.2)
\end{aligned}
$$

Through this, and analogous formulae which increment the momentum on centers **B**, **C**, or **D**, any $(\mathbf{ab}|\mathbf{cd})$ can be efficiently reduced to ${}_{a'b'p'}(\mathbf{00}|\mathbf{00})^{(m)}_{c'd'q'} \equiv {}_{a'b'p'}(m)_{c'd'q'}$ integrals. One of the most appealing features of (3.2) and its analogues is that the *only* geometric variables that appear are the Cartesian components of the vectors **B**–**A**, **C**–**D**, and **D**–**B**. This permits the development of the attractive tensor formulation for the L_2 step, which we now introduce.

Despite its complexity, (3.2) yields compact expressions for simple ERIs. For example,

$$
\begin{aligned}
(p_x s|ss) = {} & BAx\{{}_{102}(1)_{000} + {}_{011}(0)_{000}\} + CDx\ {}_{001}(1)_{101} \\
& + DBx\ {}_{001}(1)_{000}, \qquad (3.3)
\end{aligned}
$$

where, for conciseness, we have used $BAx \equiv B_x - A_x$, etc. In real applications, however, our target is inevitably a complete class of ERIs, not just one of them. Since the formulae for $(p_y s|ss)$ and $(p_z s|ss)$ are analogous to (3.3), the full $(ps|ss)$ class **I** can be concisely written as

$$I_\mu = T_{i\mu} O_i, \qquad (3.4)$$

where we have employed the Einstein summation convention and we have introduced

$$\mathbf{T} = \begin{bmatrix} BAx & BAy & BAz \\ CDx & CDy & CDz \\ DBx & DBy & DBz \end{bmatrix}, \qquad (3.5)$$

$$\mathbf{O} = [{}_{102}(1)_{000} + {}_{011}(0)_{000} \quad {}_{001}(1)_{101} \quad {}_{001}(1)_{000}]. \qquad (3.6)$$

It is easy to see that (3.4) involves 9 multiplies + 6 adds = 15 flops (floating-point operations). Equation (3.5) requires a further 3 flops if its bottom row is not already precomputed.

Tensor equations for higher ERI (and ERI derivative) classes are similar and offer a new approach in which angular momentum is built up, not by the use of single recurrence relations, but by consecutive tensor multiplications by **T**, each of which adds a unit of angular momentum (or a derivative order) to the evolving class. We will term a tensor **O** of linear combinations of ${}_{a'b'p'}(m)_{c'd'q'}$ a *kernel*, and we note that the physical interpretation of kernels is an interesting topic.[33] We will assume, throughout this paper, that tensors are multiplied from right to left.

The tensor equation for a $(ps|ps)$ or $(pp|ss)$ class is given by

$$I_{\mu\nu} = T_{i\mu} T_{j\nu} O^{(1)}_{ij} + \delta_{\mu\nu} O^{(2)}. \qquad (3.7)$$

It involves 93 flops and two kernels. In the case of $(ps|ps)$, the kernels are given by

$$\mathbf{O}^{(1)} = \begin{bmatrix} {}_{203}(2)_{001} + {}_{112}(1)_{001} & {}_{102}(2)_{102} + {}_{102}(1)_{011} + {}_{011}(1)_{102} + {}_{011}(0)_{011} & {}_{102}(2)_{001} + {}_{011}(1)_{001} \\ {}_{102}(2)_{102} & {}_{001}(2)_{203} + {}_{001}(1)_{112} & {}_{001}(2)_{102} \\ {}_{102}(2)_{001} & {}_{001}(2)_{102} + {}_{001}(1)_{001} & {}_{001}(2)_{001} \end{bmatrix}, \qquad (3.8)$$

$$\mathbf{O}^{(2)} = [{}_{001}(1)_{001}], \qquad (3.9)$$

whereas, for $(pp|ss)$, they are

$$\mathbf{O}^{(1)} = \begin{bmatrix} {}_{204}(2)_{000} - {}_{203}(1)_{000} + {}_{113}(1)_{000} - {}_{112}(0)_{000} & {}_{103}(2)_{101} + {}_{012}(1)_{101} & {}_{103}(2)_{000} + {}_{012}(1)_{000} \\ {}_{103}(2)_{101} - {}_{102}(1)_{101} & {}_{002}(2)_{202} & {}_{002}(2)_{101} \\ {}_{103}(2)_{000} - {}_{102}(1)_{000} & {}_{002}(2)_{101} & {}_{002}(2)_{000} \end{bmatrix}, \qquad (3.10)$$

$$\mathbf{O}^{(2)}=[_{001}(0)_{000}-{}_{002}(1)_{000}]. \tag{3.11}$$

A $(pp|ps)$ class can be formed in 477 flops from a third-order and three first-order kernels,

$$I_{\mu\nu\lambda}=T_{i\mu}T_{j\nu}T_{k\lambda}O^{(1)}_{ijk}+\delta_{\nu\lambda}T_{i\mu}O^{(2)}_{i}+\delta_{\mu\lambda}T_{j\nu}O^{(3)}_{j} + \delta_{\mu\nu}T_{k\lambda}O^{(4)}_{k}. \tag{3.12}$$

A $(pp|pp)$ class can be formed in 2349 flops from a fourth-, six second- and three zeroth-order kernels,

$$\begin{aligned}I_{\mu\nu\lambda\sigma}=&T_{i\mu}T_{j\nu}T_{k\lambda}T_{l\sigma}O^{(1)}_{ijkl}+\delta_{\lambda\sigma}T_{i\mu}T_{j\nu}O^{(2)}_{ij}\\&+\delta_{\nu\sigma}T_{i\mu}T_{k\lambda}O^{(3)}_{ik}+\delta_{\nu\lambda}T_{i\mu}T_{l\sigma}O^{(4)}_{il}\\&+\delta_{\mu\sigma}T_{j\nu}T_{k\lambda}O^{(5)}_{jk}+\delta_{\mu\lambda}T_{j\nu}T_{l\sigma}O^{(6)}_{jl}\\&+\delta_{\mu\nu}T_{k\lambda}T_{l\sigma}O^{(7)}_{kl}+\delta_{\mu\sigma}\delta_{\nu\lambda}O^{(8)}+\delta_{\mu\lambda}\delta_{\nu\sigma}O^{(9)}\\&+\delta_{\mu\nu}\delta_{\lambda\sigma}O^{(10)}.\end{aligned} \tag{3.13}$$

Classes containing shells with higher angular momentum are analogous. For example, $(ds|ps)$ can be formed using (3.12), but with the added simplification that only $\mu\leqslant\nu$ need be treated. With a little practice, one can quickly write down the tensor equation for any integral class.

We note that, although we have written the tensor equations above in an expanded form, we have done so only for clarity. Elementary algebraic manipulations [for example, factorizing $T_{i\mu}$ out of the first two terms in (3.12)] generally yield equations of significantly lower cost, and this is particularly worthwhile for classes with high angular momentum.

Q-Chem contains one routine to compute general binary tensor products (e.g., $T_{l\sigma}O_{ijkl}$) and another to perform the Kronecker-adds. The former, which is generally responsible for roughly 90% of the flop count, is simply a matrix multiply (a BLAS-3 construct) and runs very efficiently. An advantage of the tensor approach is that the esoteric tree-search problems[24–26] that vex other schemes reduce to the well-studied problem[34] of efficient matrix multiplication.

Having developed an elegant and efficient scheme for transforming kernels into integrals, we now turn our attention to the generation of the kernels themselves. Although these can be formed by contraction of the $[\mathbf{0}]^{(m)}$ integrals (the OC choice), it can be more efficient to form them directly from contracted Gaussians (the CO choice). We now discuss this in detail.

IV. THE O_2 STEP

We again begin with the identity (3.1)

$$\mathbf{R}\equiv(\mathbf{B}-\mathbf{A})x+(\mathbf{C}-\mathbf{D})y+(\mathbf{D}-\mathbf{B}), \tag{4.1}$$

where $x=\alpha/\zeta$ and $y=\gamma/\eta$. Taking the dot product of (4.1) with itself, one can show that

$$\frac{1}{R^{2m+1}}=\frac{1}{DB^{2m+1}}[1+S_1x+S_2y+S_3x^2+S_4xy+S_5y^2]^{-(2m+1)/2}, \tag{4.2}$$

where the S_n are simple, dimensionless functions of **A**, **B**, **C**, and **D**. If the bra and ket charge distributions are physically well separated, it is possible to expand (4.2) as the double infinite series in x and y

$$\frac{1}{R^{2m+1}}=\frac{1}{DB^{2m+1}}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}h^{(m)}_{ij}x^iy^j. \tag{4.3}$$

It is tedious but straightforward to prove inductively that the $h^{(m)}_{ij}$ can be generated recursively from the initial value $h^{(m)}_{00}\equiv1$ using

$$\begin{aligned}i(DB\cdot DB)h^{(m)}_{ij}=&(2i+2j+2m-1)(AB\cdot DB)h^{(m)}_{(i-1)j}\\&+i(DC\cdot DC)h^{(m)}_{i(j-2)}-(i+2j+2m-1)\\&\times(AB\cdot AB)h^{(m)}_{(i-2)j}-(2j+2m-1)\\&\times(AB\cdot DC)h^{(m)}_{(i-1)(j-1)}\end{aligned} \tag{4.4}$$

and

$$\begin{aligned}j(DB\cdot DB)h^{(m)}_{ij}=&(2i+2j+2m-1)(DC\cdot DB)h^{(m)}_{i(j-1)}\\&+j(AB\cdot AB)h^{(m)}_{(i-2)j}-(2i+j+2m-1)\\&\times(DC\cdot DC)h^{(m)}_{i(j-2)}-(2i+2m-1)\\&\times(AB\cdot DC)h^{(m)}_{(i-1)(j-1)}.\end{aligned} \tag{4.5}$$

Now consider the task of forming a given ${}_{a'b'p'}(m)_{c'd'q'}$. Still assuming that the bra and ket are well separated, the formula for such an integral reduces to the classical expression[10]

$$\begin{aligned}&{}_{a'b'p'}(m)_{c'd'q'}\\&=(2m-1)!!\sum_{p=1}^{K_{\text{bra}}}\sum_{q=1}^{K_{\text{ket}}}\frac{(2\alpha_p)^{a'}(2\beta_p)^{b'}}{(2\zeta_p)^{p'}}\\&\quad\times\left(\frac{U_pV_q}{R^{2m+1}_{pq}}\right)\frac{(2\gamma_q)^{c'}(2\delta_q)^{d'}}{(2\eta_q)^{q'}}.\end{aligned} \tag{4.6}$$

The cost of (4.6), which embodies the OC choice, is proportional to K_{tot}. Because it involves K_{tot} square roots and divides, (4.6) is frequently the most computationally demanding of all the steps in Fock formation, even if the bra and ket scalings are precomputed. However, if we substitute (4.3) into (4.6) and interchange the summation order (that is, use the associativity of matrix multiplication), we obtain the O_2 matrix equation

$${}_{a'b'p'}(m)_{c'd'q'}=\frac{(2m-1)!!}{DB^{2m+1}}\mathbf{u}^t\mathbf{H}\mathbf{v}, \tag{4.7}$$

where $\mathbf{H}=[h^{(m)}_{ij}]$ and **u** and **v** contain contracted Gaussian amplitudes, *viz.*

$$u_i=\sum_{p=1}^{K_{\text{bra}}}\frac{(2\alpha_p)^{a'+i}(2\beta_p)^{b'}}{(2\zeta_p)^{p'+i}}U_p, \tag{4.8}$$

$$v_j=\sum_{q=1}^{K_{\text{ket}}}\frac{(2\gamma_q)^{c'+j}(2\delta_q)^{d'}}{(2\eta_q)^{q'+j}}V_q. \tag{4.9}$$

In order to generate the full set of $_{a'b'p'}(m)_{c'd'q'}$ as efficiently as possible, we maximally factorize (4.7) by forming only the *unique* **Hv** and subsequently contracting these with the necessary **u**. Both of these steps can be accomplished using a mixture of BLAS-2 and BLAS-3 constructs.

If the contracted Gaussians (4.8) and (4.9) are precomputed (the C_1 step), the cost of (4.7) is independent of K_{tot}. Consequently, for sufficiently contracted and well-separated shell quartets, the CO choice must become more efficient than OC. The point at which this occurs depends on several factors. The latter involves many divide and square roots while the former necessitates the construction of the **H** matrix and the computation of (4.7). Which of these is cheaper depends on machine-specific computational characteristics such as the divide-to-multiply and square-root-to-multiply ratios. Moreover, the number of terms in (4.3) that are included (which determines the dimension of **H**) is also important. We have found that using all terms of ninth degree and lower (e.g., x^4y^5) affords a useful compromise. Finally, we also note that major simplifications in (4.4) and (4.5) result whenever $\mathbf{A}=\mathbf{B}$ or $\mathbf{C}=\mathbf{D}$.

V. THE *DL* STEP

The accumulation of Fock (Coulomb or exchange) contributions **F** from ERIs,

$$F_{\mu\nu}\leftarrow F_{\mu\nu}+I_{\mu\nu\lambda\sigma}P_{\lambda\sigma}, \tag{5.1}$$

(where **I** is an ERI tensor and **P** contains density matrix elements) is an important operation: two such sums are required when computing only Coulomb effects, six when treating both Coulomb and closed-shell exchange effects, and ten for Coulomb and open-shell exchange. Traditionally, **I** is computed and then contracted with **P** according to (5.1). However, Ahmadi and Almlöf[35] and White and Head-Gordon[36] have recently shown that an $OLCD$ method is often improved by introducing the density earlier to yield an $ODLC$ scheme, particularly when the degree of contraction is not too large. Within such schemes, the formation of **I** is entirely bypassed and large computational savings can result. One may reasonably infer from this that there could exist classes for which a DL path ($CODL$ or $OCDL$) is superior to the corresponding LD path ($COLD$ or $OCLD$) and, indeed, this is found to be true. We use the $(ps|ps)$ and $(pp|ps)$ classes to illustrate the point.

Consider the evaluation of the Coulomb-only contributions for a $(ps|ps)$ class. The LD scheme costs $93+18+18=129$ flops and may be conveniently summarized as

$$I_{\mu\lambda}=T_{i\mu}T_{k\lambda}O_{ik}^{(1)}+\delta_{\mu\lambda}O^{(2)}, \tag{5.2}$$

$$J_{\mu}^{\text{bra}}\leftarrow J_{\mu}^{\text{bra}}+I_{\mu\lambda}P_{\lambda}^{\text{ket}}, \tag{5.3}$$

$$J_{\lambda}^{\text{ket}}\leftarrow J_{\lambda}^{\text{ket}}+I_{\mu\lambda}P_{\mu}^{\text{bra}}. \tag{5.4}$$

However, associatively rearranging these equations yields the $54+54=108$-flop DL scheme

$$J_{\mu}^{\text{bra}}\leftarrow J_{\mu}^{\text{bra}}+T_{i\mu}O_{ik}^{(1)}T_{k\lambda}P_{\lambda}^{\text{ket}}+O^{(2)}P_{\mu}^{\text{ket}}, \tag{5.5}$$

$$J_{\lambda}^{\text{ket}}\leftarrow J_{\lambda}^{\text{ket}}+T_{k\lambda}O_{ik}^{(1)}T_{i\mu}P_{\mu}^{\text{bra}}+O^{(2)}P_{\lambda}^{\text{bra}}, \tag{5.6}$$

in which, following our convention, the **TOTP** products are evaluated from right to left. The critical advantage of DL over LD is that the matrix–matrix multiplies in (5.2) are replaced by the matrix–vector multiplies in (5.5) and (5.6). Although the 16% savings achieved for $(ps|ps)$ is modest, the savings grow with the angular momentum of the class.

The evaluation of the Coulomb-only contributions for a $(pp|ps)$ class is similar. The LD scheme costs $477+54+54=585$ flops and is summarized by

$$I_{\mu\nu\lambda}=T_{i\mu}T_{j\nu}T_{k\lambda}O_{ijk}^{(1)}+\delta_{\nu\lambda}T_{i\mu}O_{i}^{(2)}+\delta_{\mu\lambda}T_{j\nu}O_{j}^{(3)} + \delta_{\mu\nu}T_{k\lambda}O_{k}^{(4)}, \tag{5.7}$$

$$J_{\mu\nu}^{\text{bra}}\leftarrow J_{\mu\nu}^{\text{bra}}+I_{\mu\nu\lambda}P_{\lambda}^{\text{ket}}, \tag{5.8}$$

$$J_{\lambda}^{\text{ket}}\leftarrow J_{\lambda}^{\text{ket}}+I_{\mu\nu\lambda}P_{\mu\nu}^{\text{bra}}. \tag{5.9}$$

However, rearrangement yields the $248+201=449$-flop DL scheme

$$J_{\mu\nu}^{\text{bra}}\leftarrow J_{\mu\nu}^{\text{bra}}+T_{i\mu}T_{j\nu}O_{ijk}^{(1)}T_{k\lambda}P_{\lambda}^{\text{ket}}+T_{i\mu}O_{i}^{(2)}P_{\nu}^{\text{ket}} + T_{j\nu}O_{j}^{(3)}P_{\mu}^{\text{ket}}+T_{k\lambda}O_{k}^{(4)}\delta_{\mu\nu}P_{\lambda}^{\text{ket}}, \tag{5.10}$$

$$J_{\lambda}^{\text{ket}}\leftarrow J_{\lambda}^{\text{ket}}+T_{k\lambda}O_{ijk}^{(1)}T_{i\mu}T_{j\nu}P_{\mu\nu}^{\text{bra}}+T_{i\mu}O_{i}^{(2)}P_{\mu\lambda}^{\text{bra}} + T_{j\nu}O_{j}^{(3)}P_{\lambda\nu}^{\text{bra}}+T_{k\lambda}O_{k}^{(4)}\ \text{Tr}[P_{\mu\nu}^{\text{bra}}]. \tag{5.11}$$

In order to execute the DL step with maximum efficiency, one must solve a tree-search problem to determine the optimal order in which to perform the required multiplications and to identify any common intermediate quantities [such as $T_{i\mu}O_{i}^{(2)}$ in (5.10) and (5.11) above]. Furthermore, like LD, the DL step benefits from judicious factorization.

In addition to the advantage that DL may enjoy over LD in terms of flop-count, we find that DL is also often substantially superior on mop-count[37] and White and Head-Gordon have made parallel observations[36] in their recent work. This is particularly important on computers where cache misses are expensive, and is reflected in some of the timings in the next section.

VI. THEORETICAL PERFORMANCE ANALYSIS

An exhaustive theoretical characterization of the COLD PRISM would include a study of its performance in constructing a variety of types of matrix element (e.g., Coulomb, exchange, etc.) from a wide variety of shell-quartet classes (e.g., $(ss|ss)$, $(fd|ps)$, etc.) with a variety of bra and ket degrees of contraction. However, a considerable amount can be learned from a much more limited survey and, in the interests of brevity, we decided to limit the scope of the present discussion to a consideration of the construction of Coulomb matrix contributions from a small sample of shell-quartet classes using the $CODL$, $COLD$, $OCDL$, $OCLD$, and $OLCD$ paths. (We have not investigated $CDOL$ in the present study.) For comparison purposes, we have also included data for the HGP-plus-digestion ($HGPD$) path from the HGP-PRISM.[10]

Since any necessary shell-pair data can be precomputed and stored before entering the main loop over shell quartets,

TABLE I. Cost parameters[a,b] for various PRISM paths and ERI classes.

	$(ss\|ss)$			$(ps\|ps)$			$(pp\|pp)$		
Path	x	y	z	x	y	z	x	y	z
$CODL$	0	0	565[c]	0	0	2483[c]	0	0	11 521[c]
$COLD$	0	0	565[c]	0	0	2504[c]	0	0	12 360[c]
$OCDL$	8[d]	0	4	26[d]	28	115	94[d]	296	2028
$OCLD$	8[d]	0	4	26[d]	28	136	94[d]	296	2867
$HGPD$	8[d]	0	4	75[d]	0	36	665[d]	0	648
$OLCD$	8[d]	0	4	75[d]	0	36	895[d]	0	324

	$(ds\|ds)$			$(dp\|dp)$			$(dd\|dd)$		
Path	x	y	z	x	y	z	x	y	z
$CODL$	0	0	8772[c]	0	0	—	0	0	—
$COLD$	0	0	9222[c]	0	0	61 017[c]	0	0	368 616[c]
$OCDL$	82[d]	198	1043	254[d]	1482	—	586[d]	5962	—
$OCLD$	82[d]	198	1493	254[d]	1482	28 439	586[d]	5962	259 415
$HGPD$	502[d]	0	144	3158[d]	0	2520	14600	0	15 234
$OLCD$	502[d]	0	144	5336[d]	0	1296	17000	0	5184

[a]For forming **J** elements from data for well-separated shell-pairs.
[b]See Eq. (5.1).
[c]Plus 1 square root and 1 divide.
[d]Plus 1 square root and 2 divides.

we can ignore the costs of the C_1 and D_1 steps. For simplicity (and because it is true of most of the shell quartets in a large system), we assume that $\mathbf{A}\neq\mathbf{B}$ and $\mathbf{C}\neq\mathbf{D}$ and the bra and ket are well-separated.

It is common[10] to decompose the total number of flops (floating-point operations) needed to treat a single shell quartet into primitive, half-contracted and contracted contributions,

$$\text{Cost}=xK^4+yK^2+z, \tag{6.1}$$

where K is the degree of contraction of each of the four shells. In Table I, we compare the x, y, and z parameters of the five COLD PRISM paths and $HGPD$ (the most frequently used path in the GAUSSIAN 92 package[31]) for six important quartet classes. In calculating these parameters, we have allocated two flops (one add and one multiply) for the contraction of each primitive or half-contracted quantity. Although this overestimates the contraction work in cases requiring addition only (and cases where the degree of contraction is 1), the fact that modern computers can execute a multiply–add instruction as fast as an add justifies this counting scheme.

When it was introduced a decade ago, the HGP algorithm[16] was considered to be one in which contraction was accomplished comparatively early. Early contraction has the advantage that computational work is transferred out of the loops involving primitive quantities (x work) and into loops involving half-contracted (y work) or fully-contracted quantities (z work). Such transferral produces an algorithm that is less sensitive to the degree of contraction (number of Gaussians) in the ERI class, but the price for this is that the z work required may be quite large.

By the standards of COLD PRISM, however, HGP can be viewed as a *late*-contraction path, exceeded in this regard only by $OLCD$ (the Obara–Saika path[15]). The x values of $HGPD$ (Table I) are generally much greater than those of the OC paths and, of course, the CO paths have $x=0$. Correspondingly, the z values of $HGPD$ are very much smaller than those of the OC and CO paths.

It is interesting to make ‘‘back of the envelope’’ estimates of the value of K beyond which $COLD$ should be faster than $HGPD$ for the basic $(ss|ss)$ class. According to the data in Table I, $COLD$ requires 565 flops+1 square root+1 divide per contracted quartet and $HGPD$ 8 flops+1 square root+2 divides per primitive quartet. It follows from (6.1) that the paths will be equally expensive when the ratio of these costs is K^4. Making reasonable assumptions about the costs of square root and divide operations, one is led to the prediction that the $HGPD$ cost will exceed that of $COLD$ whenever $K\geqslant 2$. Of course, $K\geqslant 2$ for most of the s functions in most of the commonly used basis sets and this observation suggests that, at least for $(ss|ss)$ classes, $COLD$ should be very competitive with $HGPD$.

As the angular momentum L_{tot} of the class rises, the z parameters for the CO paths grow very quickly and rough comparisons in the vein of the preceding paragraph indicate that these paths will only rarely be useful for classes with $L_{\text{tot}}>4$. (It should be noted, however, that the very contracted d functions that are used in transition metal and heavy main-group calculations may benefit from these paths to somewhat higher values of L_{tot}.) However, for classes with low L_{tot} and high K, the CO paths are more efficient than any previously published algorithm.

The z parameters for the OC paths grow rather more slowly with L_{tot}, and the parameters in Table I suggest that these paths will be competitive for the intermediate classes

TABLE II. Timing ratios[a] for various PRISM paths and ERI classes.

Class	$(ss\|ss)$		$(ps\|ss)$		$(pp\|ss)$	
(K_{bra},K_{ket})	(12,12)	(6,6)	(8,12)	(8,6)	(8,12)	(8,6)
OC/CO	2.83	1.52	2.48	1.64	2.26	1.44
LD/DL	1.27		1.34		3.40	
$HGPD/CODL$	3.16	1.68	4.50	2.40	4.20	2.03

Class	$(ds\|ss)$		$(dp\|ss)$		$(dd\|ss)$	
(K_{bra},K_{ket})	(6,12)	(6,6)	(3,12)	(3,6)	(1,12)	(1,6)
OC/CO	2.07	1.38	1.07	1.00	1.00	0.98
LD/DL	1.70		2.50		2.05	
$HGPD/CODL$	3.73	1.90	1.92	1.23	1.22	0.73

Class	$(ps\|ps)$	$(pp\|ps)$	$(pp\|pp)$	$(ds\|ps)$	$(ds\|pp)$	$(dp\|ps)$	$(ds\|ds)$
(K_{bra},K_{ket})	(8,8)	(8,8)	(8,8)	(6,8)	(6,8)	(3,8)	(6,6)
OC/CO	2.03	1.76	1.76	1.85	1.08	0.96	1.01
LD/DL	2.00	4.23	2.40	2.27	2.98	3.11	2.46
$HGPD/CODL$	5.59	5.16	7.64	3.98	3.15	1.93	2.66

[a]For forming the HFS/6-31G* **J** matrix for an array of 340 carbon atoms (see text).

with moderate L_{tot} and K values. In order to gain the maximum benefit from these paths, however, it is critical that the subroutine that performs the x work be written very carefully in order to avoid being mop-bound,[37] i.e., limited by memory traffic rather than flops. Our implementation in Q-Chem achieves this.

The DL paths are cheaper than their LD analogs for all of the classes in Table I except $(ss|ss)$. This is not always the case, however. In particular, any class containing the $|ss)$ ket is never more cheaply treated by DL than LD. Furthermore, if exchange (rather than Coulomb) matrix elements are to be formed, the DL paths become much less competitive. As one would expect, analogous DL and LD paths have equal x and y parameters and differ only in z work. Consequently, differences between them are *rigorously independent of* K and tend to be small. This is in interesting contrast with the results of White and Head-Gordon, who found that their J matrix engine[36] (which we would classify as $OLDC$) gave huge improvements over $HGPD$.

Before moving to the empirical comparison of the next section, it is worth emphasizing again the information that Table I does *not* provide. As many authors have pointed out, the flop-count of an algorithm is only a ''zeroth-order'' indicator of an algorithm's practical utility, for it ignores the critically important question of how the algorithm is implemented. Using the computer's architecture effectively is no less important than minimizing the number of flops that are required. Although the multitude of architectures (RISC, vector, parallel, etc.) now in common use makes this a daunting prospect, all significant modern machines are designed to perform BLAS-2 and BLAS-3 arithmetic efficiently. Thus, the fact that the new COLD PRISM paths ($CODL$, $COLD$, $OCDL$, and $OCLD$) are all expressed in terms of BLAS-2 and BLAS-3, gives them a major advantage over those ($HGPD$ and $OLCD$) that cannot be so expressed. Furthermore (as we noted in Sec. V), for a given class, the DL step generally requires less memory than the LD step and, where cache effects are important, this is likely to enhance the performance of DL.

VII. EMPIRICAL PERFORMANCE ANALYSIS

Because our principal interest lies in applications to large systems, we have investigated the comparative empirical performances of COLD PRISM and $HGPD$ on such a system. The ''molecule'' that we have considered is C_{340} arranged as a 4×85 rectangular lattice in which adjacent atoms are 1.25 Å apart. The basis set used is $6-31\,G^*$ and this leads to a total of 5100 basis functions. We have used the Q-Chem program to compute the Coulomb matrix for this system and have inserted timing calls in order to measure the time consumed by various steps on the COLD PRISM. All calculations were performed on an IBM 43P Power PC workstation.

In Table II, we present timing data for a selection of the classes that typically arise. The degrees of contraction of the bra and ket are given in the form (K_{bra},K_{ket}). The actual timings (which are typically roughly 10^3 CPU seconds) are of little interest and are not included here. Instead, we have reported the *ratios* between the times required by various steps. Unlike the timings themselves, the ratios should be roughly independent of the system studied.

We first examine the OC/CO ratios. The values in Table II confirm qualitatively most of the predictions made in the foregoing section based on flop-counts. In particular, it is clear that CO is much faster than OC for classes with low angular momentum L_{tot} and high total degree of contraction $K_{tot}=K_{bra}K_{ket}$, but that the margin diminishes as L_{tot} grows and K_{tot} decreases. CO is nearly three times as fast as OC for $(ss|ss)$ with $K_{bra}=K_{ket}=12$, but is actually slightly slower for $(dd|ss)$ with $K_{bra}=1$ and $K_{ket}=6$.

The LD/DL ratios are surprisingly high in light of the flop-counts in Table I. Although DL often requires fewer flops than LD, the empirical margins in Table II are significantly larger than one would have anticipated and reflect the cache effects mentioned earlier. The ratios that we obtain are consistent with those resulting from White and Head-Gordon's J matrix engine. The first six classes in Table II have $|ss)$ kets and, according to flop-count, should be slower by DL than by LD. Yet we find that they are sometimes considerably faster!

The third set of ratios in Table II reflect the total times required by $HGPD$ and $CODL$ to form Coulomb matrix contributions from shell-pair data. Given the x, y, and z parameters from Table I for $CODL$ and $HGPD$, one would expect that $CODL$ would be the preferred path for classes with low L_{tot} and high K_{tot} and this is certainly observed to be the case. It is perhaps surprising, however, that $CODL$ is effective even for moderate-L_{tot} classes such as $(pp|pp)$ where, with $K_{\text{bra}} = K_{\text{ket}} = 8$, $CODL$ is more than seven times as fast as $HGPD$. In fact, $CODL$ is inferior to $HGPD$ for only one of the classes in Table II, namely, $(dd|ss)$ with $K_{\text{bra}} = 1$ and $K_{\text{ket}} = 6$. Of course, $HGPD$ becomes far more efficient than $CODL$ for higher classes such as $(dd|dd)$, but such classes tend not to be particularly demanding anyway in typical calculations.

VIII. CONCLUDING REMARKS

The COLD PRISM is the most general scheme hitherto developed for the construction of two-electron matrix elements from shell-pair data. We have shown that the four essential steps involved (C, O, L, and D) can be executed in several different orders and that each order yields a new algorithm. In this way, it unites the PRISM (Refs. 10 and 11) methodologies (which include $OLCD$ and $OCLD$ paths) and the J engine[36] (an $OLDC$ path) and augments them with an array of other useful paths such as $CODL$, $COLD$, and $OCDL$. Moreover, all of the new paths can be coded in terms of BLAS-2 and BLAS-3 constructs. Finally, we note that calculations of molecular electrostatic potentials using COLD PRISM have been submitted for publication elsewhere.[38]

ACKNOWLEDGMENTS

We are grateful to Chris White and Martin Head-Gordon for fruitful discussions on the DL step and we thank them, Professor Almlöf, and Professor Ishida for preprints of Refs. 36, 35, and 23. We appreciate research support from Q-Chem Inc. and the New Zealand Lottery Grants Board. TRA thanks Q-Chem Inc. and Massey University for a Postdoctoral Fellowship.

[1] W. J. Hehre, L. Radom, J. A. Pople, and P. v. R. Schleyer, *Ab Initio Molecular Orbital Theory* (Wiley, New York, 1986).
[2] L. A. Curtiss, J. E. Carpenter, K. Raghavachari, and J. A. Pople, J. Chem. Phys. **96**, 9030 (1992).
[3] C. A. White, B. G. Johnson, P. M. W. Gill, and M. Head-Gordon, Chem. Phys. Lett. **230**, 8 (1994).
[4] J. P. Dombroski, S. W. Taylor, and P. M. W. Gill, J. Phys. Chem. **100**, 6272 (1996).
[5] R. D. Adamson, J. P. Dombroski, and P. M. W. Gill, Chem. Phys. Lett. **254**, 329 (1996).
[6] P. M. W. Gill, R. D. Adamson, and J. A. Pople, Mol. Phys. **88**, 1005 (1996).
[7] R. D. Adamson and P. M. W. Gill, J. Mol. Struct. (Theochem) (in press).
[8] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory* (McGraw-Hill, New York, 1989).
[9] S. F. Boys, Proc. Soc. London Ser. A **200**, 542 (1950).
[10] P. M. W. Gill, Adv. Quantum Chem. **25**, 141 (1994).
[11] P. M. W. Gill and J. A. Pople, Int. J. Quantum Chem. **40**, 753 (1991).
[12] J. A. Pople and W. J. Hehre, J. Comp. Phys. **27**, 161 (1978).
[13] M. Dupuis, J. Rys, and H. F. King, J. Chem. Phys. **65**, 111 (1976).
[14] L. E. McMurchie and E. R. Davidson, J. Comp. Phys. **26**, 218 (1978).
[15] S. Obara, and A. Saika, J. Chem. Phys. **84**, 3963 (1986).
[16] M. Head-Gordon and J. A. Pople, J. Chem. Phys. **89**, 5777 (1988).
[17] S. Ten-no, Chem. Phys. Lett. **211**, 259 (1993).
[18] P. M. W. Gill, B. G. Johnson, and J. A. Pople, Int. J. Quantum Chem. **40**, 745 (1991).
[19] J. Rys, M. Dupuis, and H. F. King, J. Comp. Chem. **4**, 154 (1983).
[20] T. P. Hamilton and H. F. Schaefer, III, Chem. Phys. **150**, 163 (1991).
[21] R. Lindh, U. Ryu, and B. Liu, J. Chem. Phys. **95**, 5889 (1991).
[22] T. Helgaker and P. R. Taylor, Theor. Chim. Acta **83**, 177 (1992).
[23] K. Ishida, Int. J. Quantum Chem. **59**, 209 (1996).
[24] B. G. Johnson, P. M. W. Gill, and J. A. Pople, Int. J. Quantum Chem. **40**, 809 (1991).
[25] U. Ryu, Y. S. Lee, and R. Lindh, Chem. Phys. Lett. **185**, 562 (1991).
[26] B. G. Johnson, P. M. W. Gill, and J. A. Pople, Chem. Phys. Lett. **206**, 229 (1993).
[27] I. Panas and J. Almlöf, Int. J. Quantum Chem. **42**, 1073 (1992).
[28] C. L. Dawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, ACM Trans. Math. Soft. **5**, 308 (1979).
[29] J. J. Dongarra, J. du Croz, S. Hammarling, and R. J. Hanson, ACM Trans. Math. Soft. **14**, 1 (1988).
[30] J. J. Dongarra, J. du Croz, I. S. Duff, and S. Hammarling, ACM Trans. Math. Soft. **16**, 1 (1990).
[31] 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.
[32] B. G. Johnson, P. M. W. Gill, M. Head-Gordon, C. A. White, D. R. Maurice, R. D. Adamson, T. R. Adams, and N. Oumi, Q-Chem, Q-Chem Inc., Pittsburgh, PA, 1996.
[33] T. R. Adams, R. D. Adamson, and P. M. W. Gill (to be published).
[34] S. Winograd, *Arithmetic Complexity of Computations* (SIAM, Bristol, 1990).
[35] G. R. Ahmadi and J. Almlöf, Chem. Phys. Lett. **246**, 364 (1995).
[36] C. A. White and M. Head-Gordon, J. Chem. Phys. **104**, 2620 (1996).
[37] M. J. Frisch, B. G. Johnson, P. M. W. Gill, D. J. Fox, and R. H. Nobes, Chem. Phys. Lett. **206**, 225 (1993).
[38] B. G. Johnson, J. W. Kyle, T. R. Adams, and P. M. W. Gill, Chem. Phys. Lett. (submitted).