MOLECULAR INTEGRALS

OVER GAUSSIAN BASIS FUNCTIONS

Peter M.W. Gill

Department of Chemistry
Carnegie Mellon University
Pittsburgh, PA 15213, USA

ADVANCES IN QUANTUM CHEMISTRY
VOLUME 25

Table of Contents

1. QUANTUM CHEMICAL PROCEDURES

The major goal of Quantum Chemistry is to obtain solutions to atomic and molecular Schrödinger equations [1]. To be useful to chemists, such solutions must be obtainable at a tolerable computational cost and must be reasonably accurate, yet devising solution methods which meet both of these requirements has proven remarkably difficult. Indeed, although very many variations have been developed over the years, almost every currently existing method can be traced to a prototype introduced within ten years of Schrödinger's seminal paper.

The most uniformly successful family of methods begins with the simplest possible n-electron wavefunction satisfying the Pauli antisymmetry principle – a Slater determinant [2] of one-electron functions $\chi_i(\mathbf{r},\omega)$ called spinorbitals. Each spinorbital is a product of a molecular orbital $\psi_i(\mathbf{r})$ and a spinfunction $\alpha(\omega)$ or $\beta(\omega)$. The $\psi_i(\mathbf{r})$ are found by the self-consistent-field (SCF) procedure introduced [3] into quantum chemistry by Hartree. The Hartree-Fock (HF) [4] and Kohn-Sham density functional (KS) [5,6] theories are both of this type, as are their many simplified variants [7–16].

In SCF methods, the n-electron Schrödinger equation is replaced by a set of n coupled integro-differential one-electron equations. The HF equations are a well-defined approximation to the Schrödinger equation and constitute the starting point for a variety of subsequent treatments [17]: the equations of KS theory are formally equivalent [18] to the original Schrödinger equation for the ground state. In both cases, the equations are highly non-linear and require iterative techniques for their solution.

Commonly, initial guesses for the molecular orbitals are obtained and these are then used to compute the potential felt by an electron in the field of the nuclei and the other electrons. The corresponding one-electron Schrödinger equation is then solved to determine another set of orbitals and the process is continued until successive sets of orbitals differ negligibly, at which point self-consistency is said to have been achieved. The most time-consuming part of this procedure is the evaluation of the *potential* which, within a basis set (Section 2), is represented by various types of integrals (Section 3). Moreover, even if we proceed beyond the HF SCF level, to correlated levels of theory, these integrals remain central to the problem of determining the energy and wavefunction [17].

The result of any quantum chemical procedure is the molecular energy, parametrically determined by the nuclear geometry. To locate equilibrium and transition structures, we usually compute the first derivatives of the energy with respect to nuclear motion [19]; harmonic vibrational frequencies can be obtained if second derivatives are available [20]; third and higher derivatives are needed [21] for higher-level studies of potential surfaces. Not surprisingly, nth-derivatives of the integrals are required to compute nth-derivatives of the energy and the efficient generation of integrals and their nth-derivatives is the focus of this Review.

2. BASIS FUNCTIONS

Because computers can represent numbers, but not functions, the molecular orbitals at each stage of the SCF procedure have to be represented by an expansion in a finite set of basis functions $\phi_i(\mathbf{r})$, $i = 1, 2, \ldots N$. If the set is mathematically complete, the result of the SCF procedure is termed the HF or KS limit: otherwise the result is dependent on the basis set used. Many types of basis funtion have been explored, and several are currently used in routine applications. However, their interrelationships and relative strengths and weaknesses are not often clarified and it may be instructive to do so here.

2.1 Slater Functions

Until the 1960's, Slater basis functions [22]

$$\phi_a^{\text{Slater}}(\mathbf{r}) \equiv (x - A_x)^{a_x} (y - A_y)^{a_y} (z - A_z)^{a_z} \exp[-\alpha|\mathbf{r} - \mathbf{A}|] \qquad (1)$$

were very popular. Like exact wavefunctions, they exhibit cusps at the nuclei and decay exponentially but their use necessitates the evaluation of integrals which are very time-consuming to compute. Although several groups have made useful progress in developing efficient algorithms for the evaluation of such integrals, explicit use of Slater basis functions is presently restricted to rather small molecules. It should be noted, however, that any Slater function can be approximated, to any desired accuracy, by a sum of Gaussian functions and the difficult Slater integrals then become relatively easy contracted Gaussian integrals (see below). This is the philosophy of the STO–nG basis sets [23]. In a similar vein, the *product* of a pair of Slater functions can also be approximated to any accuracy by a sum of Gaussians and this approach has been suggested and explored by Harris and Monkhorst [24].

2.2 Gaussian Functions

A *primitive* Gaussian function

$$\phi_a^{PGF}(\mathbf{r}) \equiv (x-A_x)^{a_x}(y-A_y)^{a_y}(z-A_z)^{a_z}\exp\left[-\alpha|\mathbf{r}-\mathbf{A}|^2\right] \quad (2)$$

has *center* $\mathbf{A} = (A_x, A_y, A_z)$, *angular momentum* $\mathbf{a} = (a_x, a_y, a_z)$, and *exponent* α. The suggestion by Boys [25] to use Gaussians as basis functions was a crucial step in the progression of quantum chemistry from a largely qualitative, to a largely quantitative, discipline. The use of a Gaussian basis set in a HF or KS calculation leads to very much simpler integrals (see below) than those which arise within a Slater basis and, although it is known [26] that more Gaussian than Slater functions are required to achieve a given basis set quality, the simplicity of Gaussian integrals more than compensates for this.

A set of primitive basis functions with the same center and exponent are known as a *primitive shell.* For example, a set of p-functions $\{p_x, p_y, p_z\}$ on an atom is termed a primitive p-shell and, if an s-function (with the same exponent) is added, the shell becomes a primitive sp-shell. The most commonly occuring shells in modern computational chemistry are s, p, sp, d and f.

2.3 Contracted Gaussian Functions

It is found that *contracted* Gaussian functions (CGFs) [27]

$$\phi_a^{CGF}(\mathbf{r}) \equiv \sum_{k=1}^{K_A} D_{ak}(x-A_x)^{a_x}(y-A_y)^{a_y}(z-A_z)^{a_z}\exp\left[-\alpha_k|\mathbf{r}-\mathbf{A}|^2\right] \quad (3)$$

where K_A is the *degree of contraction* and the D_{ak} are *contraction coefficients*, are even more computationally effective [26] than Slater functions. It is crucial to note that, although they have different contraction coefficients and exponents, all of the primitive functions in a contracted function share the same center **A** and angular momentum **a**. A set of CGFs with the same center and the same set of exponents is termed a *contracted shell* by analogy with a primitive shell (defined above).

Over the years, many contracted Gaussian basis sets have been constructed and the interested reader will find the excellent review by Davidson and Feller [26] very illuminating. As a rule, one or two CGFs are used to model each of the core atomic orbitals (*1s* for lithium to neon; *1s*, *2s* and *2p* for sodium to argon; *etc.*) and the CGFs are often highly contracted (a typical K value is 6). Each valence atomic orbital (*1s* for hydrogen and helium; *2s* and *2p* for lithium to neon; *etc.*) is generally more weakly contracted (K less than about 4). Finally, high-quality basis sets contain functions whose angular momenta are higher than that of the valence orbitals (*e.g. p* for hydrogen and helium, *d* for lithium to argon, *etc.*) and, in most cases, these functions are uncontracted (K = 1).

Two distinct classes of contracted Gaussian functions are in common use. In *general* contraction schemes, different contracted functions share the same primitive exponents (with different contraction coefficients) while, in *segmented* schemes, different contracted functions are constructed from primitive functions with different exponents. As a rule, basis functions of the former type tend to have higher degrees of contraction but the higher computational cost implied by this can be partially ameliorated by the use of algorithms which are carefully constructed to take maximum advantage of the exponent sharing. In this Review, we will confine our attention to the efficient treatment of segmented basis sets: we will extend our analysis to the generally contracted case in a future paper.

2.4 Gaussian Lobe Functions

Many of the programming complexities which arise when general contracted Gaussian functions are used disappear if *all* of the functions are constrained to be *s*-functions, *i.e.*

$$\phi_a^{\mathrm{Lobe}}(\mathbf{r}) \equiv \sum_{k=1}^{K_A} D_{ak} \exp\left[-\alpha_k |\mathbf{r}-\mathbf{A}|^2\right] \tag{4}$$

Such basis functions were advocated by a number of authors [28] on the basis of their manifest simplicity and because an array of variously centered *s* functions can mimic functions of higher angular momentum (*p*, *d*, *f*, *etc.*). However, for obvious reasons, Gaussian Lobe basis sets have to be rather large to yield useful results and become unwieldy in high angular momentum cases. They are rarely used nowadays because of the availability of highly efficient algorithms and programs which can handle CGFs of arbitrary angular momentum.

2.5 Delta Functions

Still more of the programming complexities vanish if another constraint is applied to the Gaussian basis set, namely that its exponents be infinite, which yields a basis composed entirely of Dirac delta functions

$$\phi_a^{\text{Delta}}(\mathbf{r}) \equiv \delta(\mathbf{r}-\mathbf{A}) \tag{5}$$

The set of delta functions at all points in space is mathematically complete and procedures based on these simplest of basis functions have been devised and implemented, first [29–33] for diatomics and, more recently, [34–36] for arbitrary polyatomic systems.

The manifest simplicity of delta functions is both their strength and weakness: computer programs based on them are refreshingly straightforward but, to yield results of chemical significance, delta basis sets must be large, typically thousands of functions per atom. The construction of efficient delta basis sets (*i.e.* 3-dimensional grids) remains an active area of research but, most commonly, they consist of points on concentric spheres about each atom. Most workers use the results of Lebedev [37] who has found optimal quadrature formulae for the surface of a sphere. However, agreement has not yet been reached on which spherical radii are best [38–42].

3. SURVEY OF GAUSSIAN INTEGRAL ALGORITHMS

What are these "integrals" to which we have referred? From the fact that the Schrödinger Hamiltonian contains only one- and two-electron operators, it is straightforward to show [17] that most of the matrix elements [43] which arise in computing the SCF energy and its derivatives with respect to nuclear motion can be written in terms of integrals of the general form

$$(\mathbf{ab}|\mathbf{cd}) \equiv \iint \phi_a(\mathbf{r}_1)\phi_b(\mathbf{r}_1)\, f(|\mathbf{r}_1-\mathbf{r}_2|)\, \phi_c(\mathbf{r}_2)\phi_d(\mathbf{r}_2)\, d\mathbf{r}_1 d\mathbf{r}_2 \qquad (6)$$

and their nth-derivatives with respect to displacement of the basis functions. Each of the integrations in (6) is over 3-dimensional space and, thus, the integral is 6-dimensional. The function f(x) is normally very simple – for example, $f(x) \equiv 1/x$ in the familiar case of two-electron repulsion integrals – but it suffices for our present purposes to consider a general function.

The integral (6) is based on two pairs of basis functions, one describing electron 1 and the other describing electron 2. Since there are N functions in the basis set, there are $N(N+1)/2$ distinct basis function pairs and, similarly, there are

$$N_{\text{total}} = \frac{1}{2}\left[\frac{N(N+1)}{2}\right]\left[\frac{N(N+1)}{2}+1\right] = \frac{1}{8}N(N+1)(N^2+N+2) \qquad (7)$$

distinct integrals of the form (6). A *class* of integrals (and/or their nth-derivatives) is defined as the set of all integrals associated with a *shell-quartet*. For example, a (*pp*|*pp*) class is the set of 81 (*pp*|*pp*) integrals associated with four *p*-shells each containing three *p*-functions. Because all of the integrals in a class share the same four centers and sets of exponents, their generation involves many common intermediate quantites. For this reason, it is always computationally expedient to compute integrals and their derivatives in classes rather than individually.

When large basis sets $N > 10^2$ are used, the generation of the integrals is a major computational task: in fact, in the most common quantum chemistry methods (such as direct SCF, either in the context of HF [44] or KS [45] calculations), it is rate-determining. Obviously, therefore, it is of paramount importance to devise and implement highly efficient generation algorithms. This realization has stimulated the development of a series of integral strategies [46–61] over the last two decades.

It is convenient to divide the evolution of two-electron integral methods into three generations which are distinguished from one another by the general goals which motivated their development. In the 1950's and early 1960's, the target was simply to be able to perform SCF calculations at the most primitive level. The first algorithm for Gaussian functions was outlined by Boys in his classic paper [25] on the use of such functions in SCF calculations and his methodology was subsequently developed and elaborated by a number of workers, including Shavitt [46], Taketa, Huzinaga and O-ohata [47] and Clementi and Davis [48]. Probably the most remarkable achievements of these First Generation methods were Clementi's minimal-basis SCF calculations on pyrrole, pyridine and pyrazine [49] which, while trivial by today's standards, must have used an astonishing amount of computer time in 1967. However, despite opening the door to computational quantum chemistry, these early algorithms and their implementations were both inefficient and slow.

The "axis-switch" method of Pople and Hehre (PH), which was the centerpiece of the Gaussian 70 program [50], revolutionized notions in the early 1970's of the range of chemical systems which could routinely be submitted to a HF calculation. It constitutes a Second Generation method because, unlike its predecessors, its objective was to enable quantum chemical SCF calculations to become a standard tool in the repetoire of practising chemists. To achieve this, the algorithm had to be fully optimized and the implementation carefully designed with contemporary computer architectures in mind. The PH algorithm, which was constructed principally to deal with basis functions of low angular momentum and high degree of contraction, becomes very inefficient under other conditions and this deficiency motivated the development of the Dupuis-Rys-King (DRK) [51] and McMurchie-Davidson (MD) [52] schemes in the mid-1970's. Until the mid-1980's, many computer programs included both PH for s and p functions and one of DRK and MD for higher functions (d, f, *etc.*). These Second Generation approaches enabled quantum chemists to perform high-quality SCF calculations on an enormous variety of molecular systems and paved the way for rigorous quantum chemistry to become established as a part of mainstream chemistry.

The success of the Second Generation methods soon spawned a number of new algorithms [53–61], each of which sought to improve upon the ones which had preceded it. The primary goal for these Third Generation methods has been maximum computational efficiency and the main driving force behind their development has been the desire to apply rigorous SCF methods to molecular systems with tens, hundreds, or even thousands, of atoms.

The most recent integral algorithms evolved with, and were influenced by, the advent of supercomputer technologies – if a new method cannot be "vectorized" and/or "parallelized", it faces a cool reception these days – and, of these, the Obara-Saika-Schlegel (OS) [53, 54], Head-Gordon-Pople (HGP) [55] and PRISM [61] algorithms are the most significant.

3.1 Performance Measures

Every integral method exhibits certain strengths and certain weaknesses and there is always a trade-off between the complexity of an algorithm (which is normally directly related to the difficulty of implementing it) and its computational performance. For example, the original Boys algorithm is not very difficult to understand or to code into a computer program but, as we have seen, its practical performance leaves plenty of room for improvement. At the other extreme, while the performance of the PRISM algorithm is much more satisfactory, it is also greatly more complicated conceptually and implementationally. However, since performance is ultimately more important to more people than is the difficulty of writing the underlying computer program, we will take a pragmatic stance and ignore the latter in this Review.

If we wish to compare a variety of integral algorithms, it is certainly highly desirable to be able to *quantify* their relative performances. Hitherto, three measures of performance have been proposed and used in the literature and, although related, they are distinct and it is instructive to describe each briefly at this point.

3.1.1 Flop-Cost

The ubiquitous measure of the *theoretical* performance of an integral algorithm is its Flop-cost. More precisely, this is the number of Flops which are required to form a specified class of integrals from a defined set of starting quantities and a Flop (Floating-Point Operation) is defined to be a floating-point add, subtract, multiply or divide. For example, (using values quoted by Head-Gordon and Pople [55]), the Flop-cost of computing a (*pp*|*pp*) class, *i.e.* 81 (*pp*|*pp*) integrals, where each of the p-functions is a sum of two primitive functions (*i.e.* K = 2), is 20,000 using MD but only 15,170 using HGP.

It should be re-emphasized that a Flop-cost is a *theoretical* measure and that comparisons like the one above are practically useful only if the two algorithms have been implemented equally well: a well-written MD program would undoubtedly generate a (*pp|pp*) class much more rapidly then a poorly written HGP program. Thus, establishing that a given algorithm has a small Flop-cost for a certain integral class tells us only that the *potential* exists for an implementation of that algorithm to perform well. It is prudent to view with considerable skepticism claims which are based exclusively on a purely theoretical measure like a Flop-count.

3.1.2 Mop-Cost

A newer measure of an algorithm's *theoretical* performance is its Mop-Cost which is defined exactly as the Flop-cost except that Memory Operations (Mops) are counted instead of Floating-Point Operations (Flops). A Mop is a load from, or a store to, fast memory. There are sound theoretical reasons why Mops should be a better indicator of practical performance than Flops, especially on recent computers employing vector or RISC architectures, and this has been discussed in detail by Frisch *et al.* [62]: to cut a long story short, the Mops measure is useful because, on modern computers and in contrast to older ones, memory traffic generally presents a tighter bottleneck than floating-point arithmetic.

The Mop-cost of forming an integral class depends on the intrinsic expense of the algorithm *and* on implementational detail. Consider, for example, the following two versions of a subroutine:

Version #1

```
      DO 10 I = 1,N
          A(I) = B(I) + C(I) * D(I)
10    CONTINUE
      DO 20 I = 1,N
          E(I) = A(I) + F(I)
20    CONTINUE
```

Version #2

```
      DO 10 I = 1,N
          A(I) = B(I) + C(I) * D(I)
          E(I) = A(I) + F(I)
10    CONTINUE
```

The two versions achieve the same results but the second will run faster on many computers. Both versions have Flop-costs of 3N but their Mop-costs are different, 7N and 6N, respectively. This is because, in the second version, A(I) does not need to be loaded from fast memory since, having just been produced, it will already reside in a register. This example is certainly a very simplistic one but it serves to illustrate the principle of Mop reduction.

Mop comparisons between integral programs have not yet begun to appear in the literature but these may be anticipated in the near future. As a design rule for the future, an algorithm which minimizes Mops is a better target than one which minimizes Flops. A particularly striking example of the usefulness of this approach is the recent discovery by Johnson *et al.* that a certain family of novel recurrence relations, which appear uncompetitive with older recurrence relations on the basis of Flop-cost, are exceedingly competitive in Mop-cost and in actual timings [63].

3.1.3 CPU-Time

The most popular measure of the *practical* performance of an algorithm is the amount of CPU time which a specified computer requires to complete a specified task, commonly a single iteration of a specified SCF procedure. While this is certainly a very appealing measure (it directly reflects an expense which is uppermost in the minds of most computational chemists), it will be extremely difficult for another worker to reproduce the timing unless the specifications are very complete.

It is important not only to specify precisely the computer whose CPU time has been measured, but also to record the compiler version, any non-default compiler options that were used and the operating system. Specification of the task which was performed is even more demanding. To write, for example, that the timing pertains to one cycle of a HF calculation is far from complete. It is necessary also to state clearly exactly what was timed: Was it the integral generation only? Was Fock matrix formation included? Was the diagonalization included? And, even more important, which integrals were actually computed? As we shall shortly see, most modern integral programs use sophisticated "screening" techniques to avoid computing very small integrals. Unless a clear description of the screening procedure used is given, including the cutoff threshold, any quoted CPU-time is obviously meaningless. However, provided that all of the various requirements above are met, a CPU-time measurement can be a very useful indication of the performance of an integral algorithm.

The optimal strategy to adopt when comparing algorithms is to employ as many measures as possible in the comparison. Previous reviews of integral algorithms [64–67] have been helpful in this regard, that of Hegarty and van der Velde [66] being particularly noteworthy.

3.2 Fundamental Integrals

One of the problems which plagues the two-electron integral literature is that of notation and we certainly hope that this Review does not add to the confusion. We will adopt a notation system which appears (slowly!) to be becoming the standard and will introduce it as the need arises. We will use **A**, **B**, **C** and **D** to represent the position vectors of the centers of the four basis functions in (6) and will use α, β, γ and δ to represent exponents of generic primitives within these functions.

All discussion of two-electron integrals ultimately begins by considering the special case of the integral (6) in which each of the four functions is a primitive *s*-Gaussian, *i.e.*

$$I \equiv \iint e^{-\alpha|\mathbf{r}_1-\mathbf{A}|^2} e^{-\beta|\mathbf{r}_1-\mathbf{B}|^2} f(|\mathbf{r}_1-\mathbf{r}_2|) e^{-\gamma|\mathbf{r}_2-\mathbf{C}|^2} e^{-\delta|\mathbf{r}_2-\mathbf{D}|^2} d\mathbf{r}_1 d\mathbf{r}_2 \quad (8)$$

and we will refer to this as the *Fundamental Integral*. The first step in its evaluation is to invoke the Gaussian Product Rule ([17], p. 411) which immediately reduces the integral to

$$I = G_{AB}G_{CD}\iint e^{-\zeta|\mathbf{r}_1-\mathbf{P}|^2} f(|\mathbf{r}_1-\mathbf{r}_2|) e^{-\eta|\mathbf{r}_2-\mathbf{Q}|^2} d\mathbf{r}_1 d\mathbf{r}_2 \quad (9)$$

where

$$G_{AB} = \exp\left[\frac{-\alpha\beta}{\alpha+\beta}|\mathbf{A}-\mathbf{B}|^2\right] \qquad G_{CD} = \exp\left[\frac{-\gamma\delta}{\gamma+\delta}|\mathbf{C}-\mathbf{D}|^2\right] \quad (10)$$

$$\zeta = \alpha+\beta \qquad \eta = \gamma+\delta \quad (11)$$

$$\mathbf{P} = \frac{\alpha\mathbf{A}+\beta\mathbf{B}}{\alpha+\beta} \qquad \mathbf{Q} = \frac{\gamma\mathbf{C}+\delta\mathbf{D}}{\gamma+\delta} \quad (12)$$

The reduction from (8) to (9) is a crucial simplification because it transforms a four-center (**A**, **B**, **C**, **D**) problem into a two-center (**P**, **Q**) one. Most of the difficulties associated with the use of Slater functions can be traced to the fact that a corresponding reduction is not possible for these functions.

Next, we replace each of the three factors in the integrand of (9) by its Fourier representation

$$e^{-\zeta|\mathbf{r}_1-\mathbf{P}|^2} = (2\pi)^{-3}\int\left(\frac{\pi}{\zeta}\right)^{3/2} e^{-k_1^2/4\zeta}\, e^{i\mathbf{k}_1\cdot(\mathbf{r}_1-\mathbf{P})}\, d\mathbf{k}_1 \tag{13}$$

$$f(|\mathbf{r}_1-\mathbf{r}_2|) = (2\pi)^{-3}\int \Im(k_2)\, e^{i\mathbf{k}_2\cdot(\mathbf{r}_1-\mathbf{r}_2)}\, d\mathbf{k}_2 \tag{14}$$

$$e^{-\eta|\mathbf{r}_2-\mathbf{Q}|^2} = (2\pi)^{-3}\int\left(\frac{\pi}{\eta}\right)^{3/2} e^{-k_3^2/4\eta}\, e^{i\mathbf{k}_3\cdot(\mathbf{r}_2-\mathbf{Q})}\, d\mathbf{k}_3 \tag{15}$$

Substituting (13) – (15) into (9) and re-ordering the integrations yields

$$\begin{aligned} I = {} & (2\pi)^{-3} G_{AB} G_{CD}\left(\frac{\pi^2}{\zeta\eta}\right)^{3/2} \iiint e^{-k_1^2/4\zeta - k_3^2/4\eta}\, e^{-i\mathbf{k}_1\cdot\mathbf{P} - i\mathbf{k}_3\cdot\mathbf{Q}} \Im(k_2) \\ & (2\pi)^{-3}\int e^{i\mathbf{r}_1\cdot(\mathbf{k}_1+\mathbf{k}_2)}\, d\mathbf{r}_1\ (2\pi)^{-3}\int e^{i\mathbf{r}_2\cdot(\mathbf{k}_3-\mathbf{k}_2)}\, d\mathbf{r}_2\, d\mathbf{k}_1 d\mathbf{k}_2 d\mathbf{k}_3 \end{aligned} \tag{16}$$

The fourth and fifth integrals in (16) are Fourier representations of the three-dimensional Dirac delta function, whence

$$\begin{aligned} I = {} & \frac{G_{AB}G_{CD}}{8(\zeta\eta)^{3/2}} \\ & \iiint e^{-k_1^2/4\zeta - k_3^2/4\eta - i\mathbf{k}_1\cdot\mathbf{P} - i\mathbf{k}_3\cdot\mathbf{Q}}\, \Im(k_2)\, \delta(\mathbf{k}_1+\mathbf{k}_2)\, \delta(\mathbf{k}_3-\mathbf{k}_2)\, d\mathbf{k}_1 d\mathbf{k}_2 d\mathbf{k}_3 \end{aligned} \tag{17}$$

By virtue of the "sampling" property of the delta function, the triple integral in (17) collapses to a single integral, yielding

$$I = \frac{\pi G_{AB} G_{CD}}{2(\zeta\eta)^{3/2} R^3} \int_0^\infty u \sin u \, e^{-u^2/4T} \, \mathfrak{I}(u/R) \, du \tag{18}$$

where

$$\vartheta^2 = \frac{\zeta\eta}{\zeta+\eta} \tag{19}$$

$$\mathbf{R} = \mathbf{Q} - \mathbf{P} \tag{20}$$

$$T = \vartheta^2 R^2 \tag{21}$$

Our evaluation of the Fundamental Integral cannot proceed further than (18) unless we now specify the two-electron function f(x). We are now in a position to consider some of the integral types which arise in quantum chemical calculations: overlap, kinetic-energy, electron-repulsion, nuclear-attraction and anti-coulomb.

3.2.1 The Overlap Integral

The Fundamental Overlap Integral is given by

$$I^{\text{overlap}} \equiv \int e^{-\alpha|\mathbf{r}-\mathbf{A}|^2} \, e^{-\gamma|\mathbf{r}-\mathbf{C}|^2} \, d\mathbf{r} \tag{22}$$

and the first task which arises is to write this in the form (8). To achieve this, we choose

$$\beta = \delta = 0 \tag{23}$$

$$f^{\text{overlap}}(r) \equiv \delta(r) \tag{24}$$

in order to convert a four-center problem into a two-center one and to convert a two-electron integral into a one-electron one. This simple trick enables us to treat overlap integrals on an equal footing with any other integrals which we may be able to cast in the form (8).

The Fourier transform of (24) is easily shown to be

$$\mathfrak{I}^{\text{overlap}}(k) = 1 \tag{25}$$

and the expression which is obtained if (25) is substituted into (18) is readily integrable by parts, finally yielding

$$I^{\text{overlap}} = \left[\frac{\pi}{\alpha+\gamma}\right]^{3/2} e^{-\alpha\gamma|\mathbf{A}-\mathbf{C}|^2/(\alpha+\gamma)} \tag{26}$$

which is the familiar formula for the overlap of two *s*-Gaussians.

3.2.2 The Kinetic-Energy Integral

The Fundamental Kinetic-Energy Integral is given by

$$I^{\text{kinetic}} \equiv \int e^{-\alpha|\mathbf{r}-\mathbf{A}|^2} \left[-\tfrac{1}{2}\nabla^2\right] e^{-\gamma|\mathbf{r}-\mathbf{C}|^2} \, d\mathbf{r} \tag{27}$$

where the Laplacian denotes differentiation with respect to $\mathbf{r}$. However, it is clearly equivalent to differentiate with respect to $\mathbf{C}$, from which it immediately follows that

$$I^{\text{kinetic}} = -\frac{1}{2}\left[\frac{\partial^2}{\partial C_x^2} + \frac{\partial^2}{\partial C_y^2} + \frac{\partial^2}{\partial C_z^2}\right] \int e^{-\alpha|\mathbf{r}-\mathbf{A}|^2} \, e^{-\gamma|\mathbf{r}-\mathbf{C}|^2} \, d\mathbf{r} \tag{28}$$

Thus, the Fundamental Kinetic-Energy Integral can be obtained from the second derivatives of the Fundamental Overlap Integral with respect to motion of the center C. Since we are interested in algorithms which offer nth-derivatives of two-electron integrals, the problem of generating kinetic-energy integrals and their nth-derivatives is subsumed into the problem of generating overlap integrals and their nth-derivatives.

3.2.3 The Electron-Repulsion Integral

The Fundamental Electron-Repulsion Integral is given by

$$I^{EE} \equiv \iint e^{-\alpha|\mathbf{r}_1-\mathbf{A}|^2} e^{-\beta|\mathbf{r}_1-\mathbf{B}|^2} \frac{1}{|\mathbf{r}_1-\mathbf{r}_2|} e^{-\gamma|\mathbf{r}_2-\mathbf{C}|^2} e^{-\delta|\mathbf{r}_2-\mathbf{D}|^2} d\mathbf{r}_1 d\mathbf{r}_2 \quad (29)$$

which is obviously already of the form (8) with

$$f^{EE}(r) \equiv r^{-1} \quad (30)$$

The Fourier transform of (30) is

$$\mathfrak{I}^{EE}(k) = 4\pi k^{-2} \quad (31)$$

and, upon substituting (31) into (18), we obtain

$$I^{EE} = G_{AB}G_{CD} \frac{2\pi^2}{(\zeta\eta)^{3/2}R} \int_0^\infty \frac{\sin u}{u} e^{-u^2/4T} du \quad (32)$$

which is related to the error function and is often re-written

$$I^{EE} = G_{AB}G_{CD} \frac{2\pi^{5/2}}{\zeta\eta(\zeta+\eta)^{1/2}} F_0^{EE}(T) \quad (33)$$

where

$$F_0^{EE}(T) = \int_0^1 e^{-Tu^2} du = 1 - T/3 + T^2/10 - \ldots \quad (34)$$

The efficient computation of the function (34) has been discussed in a number of papers [46, 50–52, 54, 55, 61, 68–71] and it is generally agreed that a carefully constructed interpolation scheme, such as that described in [71], is the most effective approach.

3.2.4 The Nuclear-Attraction Integral

The Fundamental Nuclear-Attraction Integral is given by

$$I^{NE} \equiv \int e^{-\alpha|\mathbf{r}-\mathbf{A}|^2} e^{-\beta|\mathbf{r}-\mathbf{B}|^2} \frac{1}{|\mathbf{r}-\mathbf{C}|} d\mathbf{r} \tag{35}$$

and, as with the Fundamental Overlap Integral, we must begin by casting this in the form (8). This can be accomplished by taking

$$\gamma = \infty \tag{36}$$

$$\delta = 0 \tag{37}$$

$$f^{NE}(r) \equiv r^{-1} \tag{38}$$

which replaces the nuclear center by a Gaussian with an extremely large exponent and thereby transforms the problem into that of the Fundamental Electron-Repulsion Integral.

3.2.5 The Anti-Coulomb Integral

Recently, we have developed a straightforward least-squares method [72] for modeling the potential of a charge distribution using a second, simpler, distribution and Kutzelnigg and coworkers have developed a promising method [73] to enhance the rate of convergence of CI-type expansions. Curiously, both of these methods are ultimately based on the Fundamental Anti-Coulomb Integral

$$I^{AC} \equiv \iint e^{-\alpha|\mathbf{r}_1-\mathbf{A}|^2} e^{-\beta|\mathbf{r}_1-\mathbf{B}|^2} |\mathbf{r}_1-\mathbf{r}_2| e^{-\gamma|\mathbf{r}_2-\mathbf{C}|^2} e^{-\delta|\mathbf{r}_2-\mathbf{D}|^2} d\mathbf{r}_1 d\mathbf{r}_2 \tag{39}$$

We note that the only feature which distinguishes (39) from (29) is the two-electron function

$$f^{AC}(r) \equiv r^{+1} \tag{40}$$

and the evaluation of the Fundamental Anti-Coulomb Integral is similar [74–76] to that of the Fundamental Electron-Repulsion Integral.

3.3 The Boys Algorithm [25]

Boys noted in his original paper that, if one differentiates the Fundamental Integral with respect to one of the coordinates of one of the centers, one obtains a primitive integral over a p and three s functions. Moreover, further differentiations lead to primitive integrals over still higher angular momentum functions. Based on this observation, Boys proposed that formulae for general primitive integrals be obtained by the repeated differentiation of the formula for the Fundamental Integral.

In the event that we wish to compute an integral (6) in which one or more of the four Gaussian basis functions is *contracted*, the Boys algorithm expresses the contracted integral as a sum of primitive integrals and then computes each of the latter using the formulae described in the foregoing paragraph.

The Boys algorithm is pedagogically useful and also serves to emphasize the highly important connection between the derivatives of Gaussian integrals and Gaussian integrals over higher angular momentum functions. However, its practical usefulness is limited by three important considerations:

(a) It rapidly becomes exceptionally tedious (and error-prone) to generate formulae by repeated differentiation of expressions such as (26) and (33);

(b) The resulting formulae are inefficient because they fail to make use of the fact that integrals in the same class share many common intermediates;

(c) The resulting formulae are inefficient because they fail to make use of the fact that the primitives in a contracted basis function all share the same center. The task of making the best possible use of this is called the Contraction Problem.

More recent integral algorithms have sought to ameliorate, to a greater or lesser degree, each of these three deficiencies. As we will see, all of these algorithms employ recurrence relations (RR's) to express integrals of high angular momentum in terms of integrals of lower angular momentum. This tactic automatically improves consideration (a) and (b) above. Addressing (c) is more difficult but the rewards are very significant: the greatest improvements in computational efficiency over the last decade have almost all resulted from new solutions to the Contraction Problem.

3.4 The Contraction Problem

Suppose that we wish to form a class of contracted integrals and that each of the basis functions is K-fold contracted, *i.e.* is a sum of K primitive functions. Then, in a straightforward method (such as the Boys algorithm), each contracted integral (**ab**|**cd**) is expressed as a sum of its component primitive integrals [**ab**|**cd**] which, in turn, are computed individually, *i.e.*

$$(\mathbf{ab}|\mathbf{cd}) = \sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{k=1}^{K}\sum_{l=1}^{K} D_{ai}D_{bj}D_{ck}D_{dl}\left[\mathbf{a}_i\mathbf{b}_j|\mathbf{c}_k\mathbf{d}_l\right] \qquad (41)$$

It is clear from (41) that the computational effort to construct the desired class of (**ab**|**cd**) integrals will rise with the fourth power of K. In fact, it is easily shown [66] that the total Flop-cost of forming the class can always be expressed as

$$\text{Flop}-\text{cost} = xK^4 + yK^2 + z \qquad (42)$$

Of course, an analogous expression for the Mop-cost also exists.

Table I: Flop-cost parameters for generating integral classes[a]

Class	Parameter	PH	MD	HGP
(*pp*|*pp*)	*x*	220	1,100	920
	y	2,300	600	30
	z	4,000	0	330
(*sp*,*sp*|*sp*,*sp*)	*x*	220	1,500	1,400
	y	2,300	1,700	30
	z	4,000	0	800
(*dd*|*dd*)	*x*	____	27,300	14,600
	y	____	24,000	30
	z	____	0	11,300
(*ff*|*ff*)	*x*	____	342,000	108,000
	y	____	383,000	30
	z	____	0	135,000

(a) Taken from [55].

The x, y and z parameters for various algorithms and various integral classes have been tabulated in a number of papers [55, 57, 59, 61] and are valuable in rationalizing the observed performances of different integral methods. In Table I, for example, we list Flop-cost parameters for the PH, MD and HGP methods to generate various integral classes. Clearly, each method is characterized by unique sets of parameters and these yield important information about the method's theoretical performance behavior. For example, the remarkably small x parameter and remarkably large z parameter for PH argue that it should be a very powerful method for generating highly contracted (*pp*|*pp*) integrals but a wasteful one for forming uncontracted ones. Similar qualitative analyses for other methods are summarized in Table II.

Table II: Algorithmic costs as a function of degree of contraction

	PH	HGP	OS	MD	DRK
Small K	High	Low	Moderate	Moderate	Moderate
Large K	Low	Moderate	High	High	High
Contraction[a]	Early	Midway	Late	Late	Late

(a) At what point in the algorithm, the contraction step occurs.

It is clear from Table II that none of the five algorithms included is the universal panacea for all integral problems. The best single method is HGP but, since typical SCF calculations on large molecules involve highly contracted, mildly contracted and weakly contracted integrals (see Section 2), a program which seeks to be near-optimal under all circumstances has to switch from one integral algorithm to another, basing its decision upon the type of integral under consideration at any moment. The Gaussian 82 [77] and Gaussian 86 [78] programs adopted this hybrid approach, employing a PH routine (Link 311) for all integrals involving only s and p functions and a DRK routine (Link 314) for any others. However, such strategies are not only arbitrary and artificial but also render the program complicated and difficult to improve.

The third row of Table II reveals that there is a very simple correlation between the performance behaviour of an algorithm and the point at which the primitive integrals are added together into contracted integrals: early-contraction methods are best suited to highly contracted integrals; late-contraction methods are best suited to weakly contracted integrals. This observation underlies the PRISM algorithm which we will discuss shortly.

3.5 The Pople-Hehre Algorithm [50]

We have seen that PH is exceptionally efficient for highly contracted classes, exhibiting x parameters (for classes involving only s and p functions) which are very much smaller than those of other, more recent, methods. How is this efficiency achieved?

Pople and Hehre showed that, given the position vectors **A**, **B**, **C** and **D** and two exponents γ and δ, there exists a unique Cartesian axis system [79] in which many primitive integrals vanish by symmetry. Moreover, because this axis system is independent of the exponents α and β, it can be used for all α,β pairs. After looping over these, the accumulated integral combinations are rotated into a second Cartesian system [80] which depends only on **A**, **B**, **C** and **D**, and the next γ,δ pair is then selected. When all γ,δ pairs have been treated, the desired integrals are finally obtained by rotating back to the original Cartesian system of the molecule.

The x parameter in (42) measures the work required to form and manipulate exclusively primitive quantities. Thus, of the steps in the PH method, only the computation and accumulation of the non-vanishing primitive integrals contributes to x and, since the unique axis system described in the foregoing paragraph was carefully designed to minimize the number of such integrals, x is correspondingly small.

The y and z parameters in (42) measure the computational effort to rotate from the first cartesian frame to the second, and from the second to the third, respectively, and to accumulate them therein. These are relatively substantial tasks and this explains the large y and z parameters in Table I for PH.

Only if the basis functions are sufficiently contracted (*i.e.* K is large enough), does the work saved by the use of special axis systems outweigh the effort which must be expended to perform the two rotations and it is interesting to determine the value of K at which PH becomes cheaper than HGP. For a (*pp*|*pp*) class, using the parameters in table I, one finds that PH is competitive with HGP even when K is as small as 2.

Notwithstanding the impressive performance of PH on integral classes involving contracted basis functions with low angular momentum, it founders when applied to uncontracted classes with high angular momentum, for example [*dd*|*dd*], because of the huge costs incurred in the two rotation steps [81]. For such classes, new techniques had to be developed and we will discuss some of these in the next few sections of this Review.

3.6 Bras, Kets and Brakets [61]

Before proceeding further, it is useful to introduce a simple but powerful notation for integrals and their nth-derivatives. All of the modern integral algorithms can be easily represented within this notation and, of course, uniform descriptions greatly facilitate any algorithmic comparisons which we may make.

In (6) we defined a general two-electron integral over the two-electron operator f. However, it has long been realized that such an equation defines an *inner product between two functions*

$$(\mathbf{ab}| \equiv \phi_a(\mathbf{r}_1)\phi_b(\mathbf{r}_1) \tag{43}$$

$$|\mathbf{cd}) \equiv \phi_c(\mathbf{r}_2)\phi_d(\mathbf{r}_2) \tag{44}$$

Thus, taking inspiration from the notation introduced by Dirac, we will refer to (**ab**| and |**cd**) as a "bra" and "ket", respectively, and to (**ab**|**cd**) as a "braket".

For the purposes of defining bras and kets, it is useful to generalize (43) and (44) substantially. Though the resulting definitions may appear complicated, it should be borne in mind that, like (43) and (44), bras and kets are simply functions of the positions of electrons 1 and 2, respectively.

We define a *primitive bra*

$$\left[\begin{matrix}\bar{a} & \bar{b} & \\ a & b & p \\ a' & b' & p'\end{matrix}\right| \equiv \frac{(2\alpha)^{a'}(2\beta)^{b'}}{(2\zeta)^{p'}} \cdot \frac{\partial^{\bar{a}+\bar{b}}}{\partial A_x^{\bar{a}_x}\partial A_y^{\bar{a}_y}\partial A_z^{\bar{a}_z}\partial B_x^{\bar{b}_x}\partial B_y^{\bar{b}_y}\partial B_z^{\bar{b}_z}} \left[\begin{matrix}0 & 0 & \\ a & b & p \\ 0 & 0 & 0\end{matrix}\right| \tag{45}$$

where

$$\left[\begin{matrix}0 & 0 & \\ a & b & p \\ 0 & 0 & 0\end{matrix}\right| \equiv$$

$$D_A D_B e^{-\alpha(\mathbf{r}-\mathbf{A})^2} e^{-\beta(\mathbf{r}-\mathbf{B})^2} \prod_{i=x,y,z} (i-A_i)^{a_i}(i-B_i)^{b_i}\zeta^{p_i/2} H_{p_i}\left[\zeta^{1/2}(i-P_i)\right] \tag{46}$$

and H_n is the nth Hermite polynomial. The other symbols in (45) and (46) have been defined earlier in this Review. The Hermite polynomials are defined by $H_0(x) \equiv 1$, $H_1(x) \equiv 2x$ and

$$H_{n+1}(x) \equiv 2x\,H_n(x) - 2n\,H_{n-1}(x) \tag{47}$$

and possess the useful property that $dH_n(x) / dx = 2n\,H_{n-1}(x)$.

Certain special cases of (45) arise sufficiently often that it is useful to introduce more concise notations for them. In particular, a Hermite function on center **P** is denoted by

$$[\mathbf{p}| \equiv \begin{bmatrix} \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{p} \\ 0 & 0 & 0 \end{bmatrix} \tag{48}$$

a product of Cartesian Gaussian primitives on **A** and **B** is

$$[\mathbf{ab}| \equiv \begin{bmatrix} \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} \\ 0 & 0 & 0 \end{bmatrix} \tag{49}$$

and the product of Cartesian Gaussian primitives on **A** and **B** and a Hermite function on **P** is

$$[\mathbf{abp}| \equiv \begin{bmatrix} \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{p} \\ 0 & 0 & 0 \end{bmatrix} \tag{50}$$

A bra in which $\bar{\mathbf{a}} = \bar{\mathbf{b}} = \mathbf{a} = \mathbf{b} = \mathbf{0}$ will be termed a *p-bra.*

Having defined a primitive bra, we define a *contracted bra* by

$$\left(\begin{matrix}\bar{\mathbf{a}} & \bar{\mathbf{b}} & \\ \mathbf{a} & \mathbf{b} & \mathbf{p} \\ \mathbf{a}' & \mathbf{b}' & \mathbf{p}'\end{matrix}\right| \equiv \sum_{k_A=1}^{K_A} \sum_{k_B=1}^{K_B} \left[\begin{matrix}\bar{\mathbf{a}} & \bar{\mathbf{b}} & \\ \mathbf{a} & \mathbf{b} & \mathbf{p} \\ \mathbf{a}' & \mathbf{b}' & \mathbf{p}'\end{matrix}\right| \tag{51}$$

and its various special cases by

$$_{\mathbf{a}'\mathbf{b}'\mathbf{p}'}(\mathbf{p}| \equiv \left(\begin{matrix}\mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{p} \\ \mathbf{a}' & \mathbf{b}' & \mathbf{p}'\end{matrix}\right| \tag{52}$$

$$(\mathbf{ab}| \equiv \left(\begin{matrix}\mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} \\ 0 & 0 & 0\end{matrix}\right| \tag{53}$$

$$_{\mathbf{a}'\mathbf{b}'\mathbf{p}'}(\mathbf{abp}| \equiv \left(\begin{matrix}\mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{p} \\ \mathbf{a}' & \mathbf{b}' & \mathbf{p}'\end{matrix}\right| \tag{54}$$

The definitions of *primitive and contracted kets* are entirely analogous to (45)–(54). As previously discussed, a braket is then an inner product between a bra and a ket

$$\left(\begin{matrix}\bar{\mathbf{a}} & \bar{\mathbf{b}} & \\ \mathbf{a} & \mathbf{b} & \mathbf{p} \\ \mathbf{a}' & \mathbf{b}' & \mathbf{p}'\end{matrix}\right|\left.\begin{matrix}\bar{\mathbf{c}} & \bar{\mathbf{d}} & \\ \mathbf{c} & \mathbf{d} & \mathbf{q} \\ \mathbf{c}' & \mathbf{d}' & \mathbf{q}'\end{matrix}\right) \equiv \iint \left(\begin{matrix}\bar{\mathbf{a}} & \bar{\mathbf{b}} & \\ \mathbf{a} & \mathbf{b} & \mathbf{p} \\ \mathbf{a}' & \mathbf{b}' & \mathbf{p}'\end{matrix}\right| f(|\mathbf{r}_1 - \mathbf{r}_2|) \left|\begin{matrix}\bar{\mathbf{c}} & \bar{\mathbf{d}} & \\ \mathbf{c} & \mathbf{d} & \mathbf{q} \\ \mathbf{c}' & \mathbf{d}' & \mathbf{q}'\end{matrix}\right) d\mathbf{r}_1 d\mathbf{r}_2$$

... (55)

and a *pq-braket*, which results from a p-bra and a q-ket, is an important special case. It should be noted that the symbols which we use to represent bras, kets and brakets, although "matrix-like" in appearance, have no connection whatever to matrices. They are, we emphasize, nothing more than compact notations for very general one-electron functions and their inner products.

3.7 The McMurchie-Davidson Algorithm [52]

Whereas the principal concern expressed in the paper by Pople and Hehre [50] was extremely high efficiency for a limited set of integral classes, the main emphasis of McMurchie and Davidson was generality and extendability. Not surprisingly, then, their respective algorithms are, to a large degree, complementary.

The target (**ab**|**cd**) are made from bra-contracted [**ab**|**cd**] using

$$(\mathbf{ab}|\mathbf{cd}) = \sum_{k=1}^{K_C}\sum_{l=1}^{K_D}\left[\mathbf{ab}\middle|\mathbf{c}_k\mathbf{d}_l\right] \tag{56}$$

MD used elementary properties of Hermite polynomials to derive a three-term RR

$$\left|(\mathbf{c}+\mathbf{1}_i)\mathbf{dq}\right] = q_i\left|\mathbf{cd}(\mathbf{q}-\mathbf{1}_i)\right] + (Q_i - C_i)\left|\mathbf{cdq}\right] + (2\eta)^{-1}\left|\mathbf{cd}(\mathbf{q}+\mathbf{1}_i)\right] \tag{57}$$

by which the [**ab**|**cd**] can be formed from [**ab**|**q**]. (Note that the subscript i represents a Cartesian direction (x, y or z) and $\mathbf{1}_i$ is the unit 3-vector in the ith direction.) Since it has no effect on the bras, we describe this step as a "ket-transformation".

The (**ab**|**q**] are formed from [**ab**|**q**] using

$$(\mathbf{ab}|\mathbf{q}] = \sum_{k=1}^{K_A}\sum_{l=1}^{K_B}\left[\mathbf{a}_k\mathbf{b}_l\middle|\mathbf{q}\right] \tag{58}$$

and, in the bra-transformation step, the [**ab**|**q**] are formed from [**p**|**q**] using the bra version

$$\left[(\mathbf{a}+\mathbf{1}_i)\mathbf{bp}\right| = p_i\left[\mathbf{ab}(\mathbf{p}-\mathbf{1}_i)\right| + (P_i - A_i)\left[\mathbf{abp}\right| + (2\zeta)^{-1}\left[\mathbf{ab}(\mathbf{p}+\mathbf{1}_i)\right| \tag{59}$$

of the RR (57).

To evaluate the [**p**|**q**], one can follow the procedure discussed in Section 3.2. This turns out to be straightforward because the Fourier Transform of a p-bra is just a Cartesian Gaussian function and it is not difficult to show that

$$[\mathbf{p}|\mathbf{q}] = (-1)^q[\mathbf{p}+\mathbf{q}] \tag{60}$$

where, for repulsion integrals,

$$[\mathbf{r}] = D_A D_B D_C D_D\, G_{AB} G_{CD}\, \frac{2\pi^{5/2}}{\zeta\eta(\zeta+\eta)^{1/2}} \int_0^1 (\vartheta u)^r H_{r_x}(R_x\vartheta u)\, H_{r_y}(R_y\vartheta u)\, H_{r_z}(R_z\vartheta u) e^{-Tu^2}\, du \tag{61}$$

which is a generalization of (33). By invoking the elementary RR for differentiation of Hermite polynomials, MD showed that $[\mathbf{r}] \equiv [\mathbf{r}]^{(0)}$ integrals can be generated from $[\mathbf{0}]^{(m)}$ integrals using the two-term RR

$$[\mathbf{r}]^{(m)} = R_i\,[\mathbf{r}-\mathbf{1}_i]^{(m+1)} - (r_i - 1)\,[\mathbf{r}-\mathbf{2}_i]^{(m+1)} \tag{62}$$

where $\mathbf{2}_i$ denotes twice the unit vector in the ith direction,

$$[\mathbf{0}]^{(m)} = D_A D_B D_C D_D\, G_{AB} G_{CD}\, \frac{2^{1/2}\pi^{5/2}}{(\zeta\eta)^{3/2}} (2\vartheta^2)^{m+1/2}\, F_m(T) \tag{63}$$

and

$$F_m(T) = \int_0^1 u^{2m} e^{-Tu^2} du \tag{64}$$

MD advocated the use of seven-term Taylor interpolation to evaluate (64) for the largest value of m, followed by downward recursion using

$$F_m(T) = \frac{1}{2m+1}\left[\, 2TF_{m+1}(T) + e^{-T} \,\right] \tag{65}$$

to compute (64) for smaller m values.

3.8 The Obara-Saika-Schlegel Algorithm [53, 54]

Four years before the appearance of the first paper by Obara and Saika, Schlegel published an important article concerning the rapid computation of *first derivatives* of two-electron integrals with respect to nuclear motion. However, although his method was in widespread use for some years, it was apparently not recognized that it is equivalent to a completely new algorithm for computing integrals themselves: only after Obara and Saika independently discovered the same algorithm did the penny drop.

The target $(\mathbf{ab}|\mathbf{cd})$ are generated from $[\mathbf{ab}|\mathbf{cd}]$ according to

$$(\mathbf{ab}|\mathbf{cd}) = \sum_{i=1}^{K_A}\sum_{j=1}^{K_B}\sum_{k=1}^{K_C}\sum_{l=1}^{K_D}\left[\mathbf{a}_i\mathbf{b}_j|\mathbf{c}_k\mathbf{d}_l\right] \tag{66}$$

and $[\mathbf{ab}|\mathbf{cd}] \equiv \langle\mathbf{ab}|\mathbf{cd}\rangle^{(0)}$ are formed from $\langle\mathbf{00}|\mathbf{00}\rangle^{(m)}$ using the eight-term OS RR

$$\begin{aligned}\langle(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}\rangle^{(m)} &= (P_i - A_i)\langle\mathbf{ab}|\mathbf{cd}\rangle^{(m)} + \frac{\eta}{\zeta+\eta}R_i\langle\mathbf{ab}|\mathbf{cd}\rangle^{(m+1)} \\ &+ \frac{a_i}{2\zeta}\left[\langle(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}\rangle^{(m)} - \frac{\eta}{\zeta+\eta}\langle(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}\rangle^{(m+1)}\right] \\ &+ \frac{b_i}{2\zeta}\left[\langle\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}\rangle^{(m)} - \frac{\eta}{\zeta+\eta}\langle\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}\rangle^{(m+1)}\right] \\ &+ \frac{c_i}{2(\zeta+\eta)}\langle\mathbf{ab}|(\mathbf{c}-\mathbf{1}_i)\rangle^{(m+1)} + \frac{d_i}{2(\zeta+\eta)}\langle\mathbf{ab}|\mathbf{c}(\mathbf{d}-\mathbf{1}_i)\rangle^{(m+1)}\end{aligned} \tag{67}$$

and its analogues which increment $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$. The $\langle\mathbf{00}|\mathbf{00}\rangle^{(m)}$, which are closely related to the $[\mathbf{0}]^{(m)}$ in (63), are defined by

$$\langle\mathbf{00}|\mathbf{00}\rangle^{(m)} = D_A D_B D_C D_D\, G_{AB} G_{CD}\, \frac{2\pi^{5/2}}{\zeta\eta(\zeta+\eta)^{1/2}}\, F_m(T) \tag{68}$$

When generating a class of integrals with non-trivial angular momentum, there are usually very many sequences in which (67) can be applied and the task of determining the most efficient sequence can be discussed as a tree-search problem. OS did not provide a solution to this problem however and, as such, the OS algorithm is not completely well-defined.

3.9 The Head-Gordon-Pople Algorithm [55]

Although no clear prescription was given by OS for the use of their RR, it was clearly a significant advance and immediately received considerable attention. The next major step forward was taken by HGP who not only introduced a second RR but also gave a precise description of how the two RR's can be used in tandem to good effect.

The target (**ab**|**cd**) are made from (**ab**|**n0**) using a two-term RR

$$\left|\mathbf{c}(\mathbf{d}+\mathbf{1}_i)\right) \equiv \left|(\mathbf{c}+\mathbf{1}_i)\mathbf{d}\right) + (\mathrm{C}_i - \mathrm{D}_i)\left|\mathbf{cd}\right) \tag{69}$$

which HGP derived from the eight-term RR of OS and which they designated the horizontal recurrence relation (HRR). In fact, the same RR (termed a transfer relation) had been used for many years to transform the 2-dimensional integrals which arise in the DRK method [51] but, until the HGP paper, it had apparently not been applied to *contracted* integrals. This distinction is important, for it implies that the computational expense incurred by the use of (69) is independent of the contraction degree of the integrals, *i.e.* the work contributes only to the z parameter in (42).

The (**ab**|**n0**) are made from (**m0**|**n0**), again using the HRR, *i.e.*

$$\left(\mathbf{a}(\mathbf{b}+\mathbf{1}_i)\right| = \left((\mathbf{a}+\mathbf{1}_i)\mathbf{b}\right| + (\mathrm{A}_i - \mathrm{B}_i)\left(\mathbf{ab}\right| \tag{70}$$

and, as before, (70) contributes only to the z parameter in (42).

The (**m0**|**n0**) are made by simple contraction of [**m0**|**n0**], *i.e.*

$$(\mathbf{m0}|\mathbf{n0}) = \sum_{i=1}^{\mathrm{K_A}} \sum_{j=1}^{\mathrm{K_B}} \sum_{k=1}^{\mathrm{K_C}} \sum_{l=1}^{\mathrm{K_D}} \left[\mathbf{m}_i\mathbf{0}_j|\mathbf{n}_k\mathbf{0}_l\right] \tag{71}$$

and the $[\mathbf{m0}|\mathbf{n0}] \equiv \langle \mathbf{m0}|\mathbf{n0}\rangle^{(0)}$ are generated using the OS RR. Because the second and fourth indices in $\langle \mathbf{m0}|\mathbf{n0}\rangle$ are **0**, the 5th, 6th, and 8th terms of the OS RR always vanish and HGP called this five-term special case the vertical recurrence relation (VRR). A simple method was given by HGP for deciding whether to reduce at the first or at the third index when applying the VRR and this completes the specification of their algorithm.

3.10 Variations on the HGP Theme

Since the advent of HGP, there have been a number of attempts to improve upon it without changing its essential structure, *viz.*

$$\langle \mathbf{00}|\mathbf{00}\rangle^{(\mathbf{m})} \rightarrow [\mathbf{m0}|\mathbf{n0}] \rightarrow (\mathbf{m0}|\mathbf{n0}) \rightarrow (\mathbf{ab}|\mathbf{cd}) \qquad (72)$$

Initially, it was suggested [82] that, instead of using the VRR, a DRK scheme (such as that of Saunders [65]) could be used to generate the **[m0|n0]**, after which the contraction and HRR steps could proceed as usual. This idea was explored by Lindh, Ryu and Liu (LRL) in a fascinating paper [59a] which convincingly argues that, for the construction of **[m0|n0]**, the HGP and DRK algorithms are essentially equivalent mathematically. This remarkable result was also supported by Flop-cost determinations, using each method, for (*pp*|*pp*), (*dd*|*dd*) and (*ff*|*ff*) classes.

A second, and somewhat more successful, attempt to render the VRR obsolete was made independently by Hamilton and Schaefer (HS) [57] and LRL [59a]. By combining the translational invariance condition for first derivatives [83] with the HRR, HS discovered a six-term RR

$$\begin{aligned}[\mathbf{ab}|(\mathbf{c}+\mathbf{1}_i)\mathbf{d}] &= \frac{a_i}{2\eta}[(\mathbf{a}-\mathbf{1}_i)\mathbf{b}|\mathbf{cd}] + \frac{b_i}{2\eta}[\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}] + \frac{c_i}{2\eta}[\mathbf{ab}|(\mathbf{c}-\mathbf{1}_i)\mathbf{d}] \\ &+ \frac{d_i}{2\eta}[\mathbf{ab}|\mathbf{c}(\mathbf{d}-\mathbf{1}_i)] - \frac{2\zeta}{2\eta}[(\mathbf{a}+\mathbf{1}_i)\mathbf{b}|\mathbf{cd}] - \left[\frac{2\beta}{2\eta}(A_i - B_i) + \frac{2\delta}{2\eta}(C_i - D_i)\right][\mathbf{ab}|\mathbf{cd}]\end{aligned} \qquad \ldots (73)$$

which, if applied to integrals in which $\mathbf{b} = \mathbf{d} = \mathbf{0}$, becomes

$$\begin{aligned}[\mathbf{m0}|(\mathbf{n}+\mathbf{1}_i)\mathbf{0}] &= \frac{m_i}{2\eta}[(\mathbf{m}-\mathbf{1}_i)\mathbf{0}|\mathbf{n0}] + \frac{n_i}{2\eta}[\mathbf{m0}|(\mathbf{n}-\mathbf{1}_i)\mathbf{0}] \\ &- \frac{2\zeta}{2\eta}[(\mathbf{m}+\mathbf{1}_i)\mathbf{0}|\mathbf{n0}] - \left[\frac{2\beta}{2\eta}(A_i - B_i) + \frac{2\delta}{2\eta}(C_i - D_i)\right][\mathbf{m0}|\mathbf{n0}]\end{aligned} \qquad \ldots (74)$$

Both HS and LRL proposed that the OS RR be used only to generate $[\mathbf{m0|00}] \equiv \langle \mathbf{m0|00} \rangle^{(0)}$ and that (74) then be used to form **[m0|n0]** from these. It seems probable from the estimates made by both groups (they disagree somewhat) of the Flop-cost of this new algorithm that it constitutes a marginal improvement over the HGP algorithm.

In the same paper where they discuss an alternate derivation of (74), LRL also present the Reduced Multiplication Rys (RMR) scheme for computing **[m0|n0]** and compare it with the DRK, HGP and HS algorithms. RMR is found to require noticeably fewer Flops than DRK and HGP (which, as mentioned above, are equivalent). The major source of this improvement is the sophisticated treatment of reusable intermediate data by RMR and the interested reader is referred to [59a] for further details.

As a general rule, the construction of the **[m0|n0]** using the VRR is considerably more expensive than the subsequent contraction and HRR steps. However, if one is dealing with integral classes of high angular momentum and low contraction or with derivatives of such classes with respect to nuclear motion, the HRR step can become sufficiently expensive to warrant optimization and Ryu, Lee and Lindh [59b] have recently studied this problem. Recognizing that efficient application of the HRR involves a complicated tree-search problem, they devised a heuristic solution which eliminates 13%, 25%, 38% and 44%, respectively, of the Flops which previous HRR implementations had needed for (*ff*|*ff*), (*gg*|*gg*), (*hh*|*hh*) and (*ii*|*ii*) classes.

Recently, however, Johnson *et al.* [63] have found that, if one wishes to minimize the number of *Mops* (as opposed to Flops) in the transformation of **(m0|n0)** to **(ab|cd)**, it is often preferable to dispense with the HRR entirely and, in lieu of it, employ RR's from a novel family which these authors term "nth-order transfer relations". We will say more about these later in the context of the PRISM algorithm.

4. THE PRISM ALGORITHM [61]

In addition to stimulating a number of variations on the HGP theme, the seminal paper by Head-Gordon and Pople [55] also served to catalyse the development of a completely different approach to the Contraction Problem called the PRISM algorithm.

We recall from our discussion of the Contraction Problem (Section 3.4) that none of the algorithms hitherto suggested has proven optimal under all circumstances: the Pople-Hehre method is highly efficient for highly contracted classes but is very poor for weakly contracted ones; the Hamilton-Schaefer-Lindh-Ryu-Liu method employing (74) is extremely effective for mildly contracted classes but is otherwise grossly inferior to Pople-Hehre. However, we observed that there is a simple connection between the behavior of an algorithm and the point at which the primitive integrals are combined to yield contracted integrals: early-contraction methods are best suited to highly contracted classes and late-contraction methods are best suited to weakly contracted classes.

Ideally, we would like an algorithm to *choose dynamically*, on the basis of the type of class being generated, the optimal point at which to perform the contraction step. The crucial insight that leads to the PRISM algorithm is the realization that some of the algorithms which we have already discussed can be generalized into forms in which such dynamic flexibity is possible. We call the generalizations of the MD and HGP methods the MD–PRISM and HGP–PRISM algorithms, respectively.

The MD–PRISM [61] was discovered and implemented long before the HGP–PRISM (which has not previously been discussed in the literature) but, for the purposes of this Review, it is convenient to develop them together in order to emphasize their similarities and differences.

We have already observed (Section 3.6) that any two-electron integral (6), and any nth-derivative of that integral with respect to motion of the basis functions, is an inner product between a function of the position of electron #1 (which we term a bra) and a function of the position of electron #2 (which we term a ket). Henceforth, rather than explicitly considering integrals and/or their nth-derivatives, we will examine the more general problem of computing contracted brakets.

After developing the mathematical foundations of the PRISM algorithm, we will discuss its implementation within the Gaussian 92 computer program [84].

The MD–PRISM and HGP–PRISM algorithms are most easily understood when presented in terms of diagrams which resemble rectangular prisms – whence their names. To simplify the discussion, we will confine our attention to the "front" face of each prism. The generalizations to the complete prisms are neither difficult nor especially interesting: they are outlined in [61c].

The front face of the MD PRISM is shown below. As the arrows indicate, the MD–PRISM algorithm consists of a set of highly interrelated pathways from shell-pair data to the desired brakets.

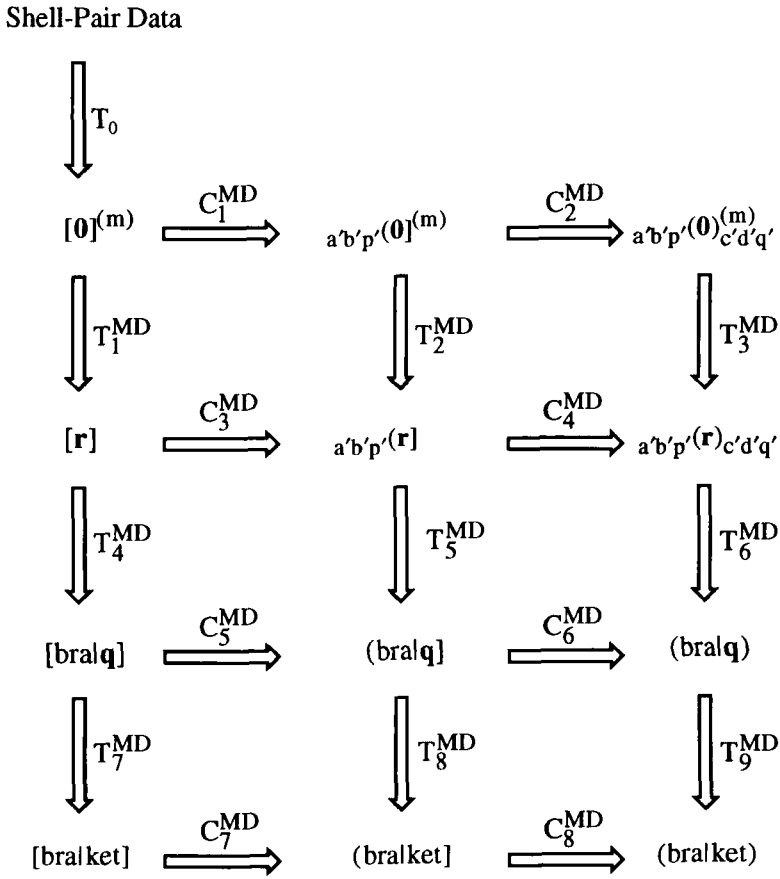

Figure 1. The front face of the MD PRISM.

The front face of the HGP PRISM is shown below. Like the MD–PRISM algorithm, the HGP–PRISM algorithm consists of a set of highly interrelated pathways from shell-pair data to the desired brakets.

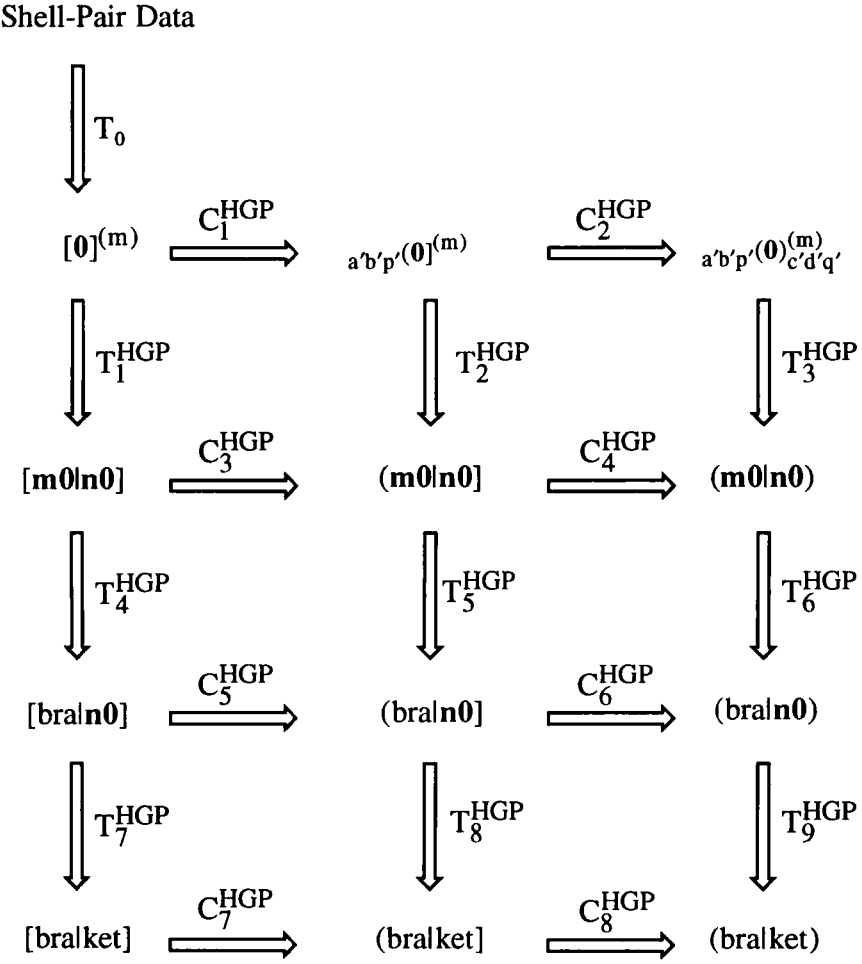

Figure 2. The front face of the HGP PRISM.

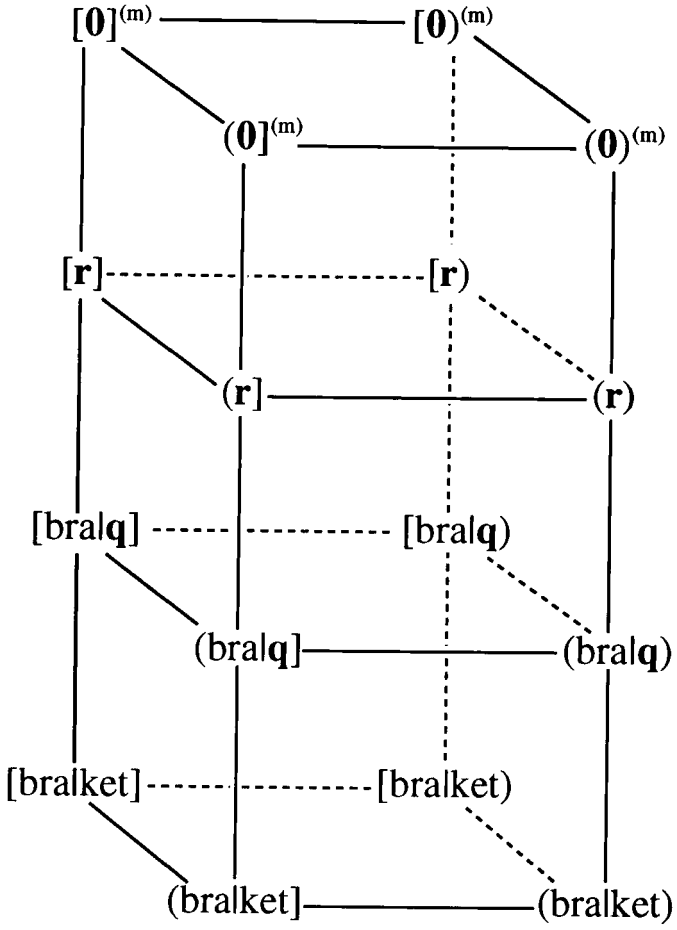

Figure 3. The Full MD PRISM.

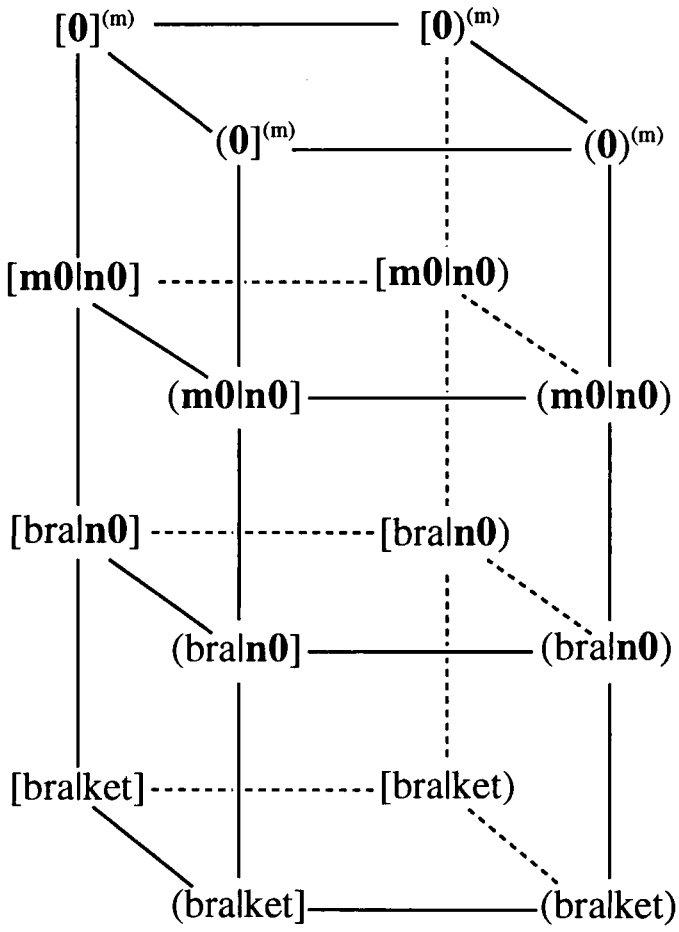

Figure 4. The Full HGP PRISM

In the subsections which follow, we will examine various aspects of the MD and HGP PRISMs but, before doing so, we may make some general observations.

(1) The first step (T_0), the generation of $[\mathbf{0}]^{(m)}$ integrals from shell-pair data, is common to both the MD– and HGP–PRISM algorithms. We will discuss T_0 in detail in Section 4.3.

(2) In either algorithm, the metamorphosis of $[\mathbf{0}]^{(m)}$ to brakets involves exactly two Contraction steps (C_i, denoted by horizontal arrows and discussed in Section 4.4) and three Transformation steps (T_i, denoted by vertical arrows and discussed in Section 4.5) and, since these five steps can be performed in *any* order, there are 10 paths from $[\mathbf{0}]^{(m)}$ to brakets in Figure 1 and a further 10 in Figure 2. We will use appropriate permutations of two "C"s and three "T"s to label these paths. For example, one of the paths in Figure 1 is MD–CCTTT and one of those in Figure 2 is HGP–TCCTT.

(3) By comparing Figures 1 and 2 with the descriptions in Sections 3.7 and 3.9, it becomes immediately apparent that the classical McMurchie–Davidson and Head–Gordon–Pople algorithms correspond to the MD–TTCTC and HGP–TCCTT paths, respectively.

(4) As a result of the "perpendicular" separation of contraction and transformation steps in Figures 1 and 2, the steps which contribute to the x, y, and z cost-parameters of (42) are clearly delineated. Specifically, the T_0, T_1, T_4, T_7, C_1, C_3, C_5 and C_7 steps contribute to x; the T_2, T_5, T_8, C_2, C_4, C_6 and C_8 steps contribute to y; and the T_3, T_6 and T_9 steps contribute to z. This observation permits a computer program to compute x, y, and z parameters (for a given braket class) for each of the paths on each of the prisms and, using (42), then very easily to determine the Flop- or Mop-cheapest path.

4.1 Shell-Pair Data

Inspection of (6) reveals that it describes the interaction of two charge distributions, $\phi_a(\mathbf{r})\phi_b(\mathbf{r})$ and $\phi_c(\mathbf{r})\phi_d(\mathbf{r})$, and our first task is to collect information about all such charge distributions in the molecule. Because brakets are formed in classes, rather than individually, it is convenient to compile data for shell-pairs (rather than basis-function-pairs) and this shell-pair dataset is central to any modern integral program. To generate all of the desired brakets, we will later loop over all pairs of shell-pairs, that is, over all shell-quartets.

We begin by considering all pairs [85] of shells in the basis set, categorizing each shell-pair as either *significant* or *negligible*. A shell-pair is negligible if the shells involved are so far apart (relative to their diffuseness) that their overlap is negligible: otherwise, it is significant. Because all of the basis functions which we have discussed (Section 2) decay at least exponentially, *most of the shell-pairs in a large molecule are negligible*. Indeed, the number of significant pairs grows only *linearly* with increasing molecular size and it therefore follows that the number of significant shell-quartets grows only *quadratically*. This is a very important point for it is this, more than anything else, which permits rigorous SCF calculations to be performed on very large systems.

Each time that we discover a significant shell-pair, we generate a *model* for that pair, *i.e.* a second shell-pair, *with fewer primitives than the first*, whose potential is as close as possible (in a least-squares sense) to that of the first. Only very recently has the theory necessary for such *potential*-fitting procedures been developed [72] and our methodology may be contrasted with previous approaches [86–92] in which many alternative modeling criteria have been employed. In particular, we note that it is common in Kohn-Sham calculations to expand the electronic density in an auxiliary basis set [89]. This is equivalent to modeling each charge distribution $\phi_a(\mathbf{r})\phi_b(\mathbf{r})$ by an expansion in the auxiliary basis. The modeling scheme employed to obtain the results discussed later in this Review was the simple one described by Head-Gordon and Pople [55]: all primitives with amplitudes below 10^{-10} are discarded while all others are kept. More elaborate modeling procedures are currently being developed [72b].

Once we have compiled a list of models for all of the significant shell-pairs, they are sorted by "type", *i.e.* by the angular momenta of the component shells and by the degree of contraction of the (modeled) shell-pair. Thus, all uncontracted *ss*-pairs are stored consecutively, followed by all doubly-contracted

(K=2) *ss*-pairs, and so forth. For each shell-pair, the parent shells and (**A** – **B**) vector are recorded; for each component primitive, we record (2α), (2β), 1/(2ζ), **P** and

$$U_P = D_A D_B G_{AB} (\pi/\zeta)^{3/2} \left[\frac{1}{2\zeta}\right]^{a+b} \tag{75}$$

The last factor in (75), which scales it according to the angular momentum (a + b) of the shell-pair, is termed the *principal scaling* and is included only if the MD–PRISM is used. For reasons which will become clearer below, its presence reduces the Flop- and Mop-costs of the algorithm.

We have recently developed [93] an upper-bound on (6)

$$(\mathbf{ab}|\mathbf{cd}) \leq \text{Min}\left\{ I^*_{ab} I^*_{cd} , \; V^*_{ab} S^*_{cd} , \; S^*_{ab} V^*_{cd} \right\} \tag{76}$$

where

$$I^*_{ab} = (\mathbf{ab}|\mathbf{ab})^{1/2} \tag{77}$$

$$V^*_{ab} = \underset{\mathbf{G}}{\text{Max}} \left| \int \phi_a(\mathbf{r}) \, \phi_b(\mathbf{r}) \, f(|\mathbf{r} - \mathbf{G}|) \, d\mathbf{r} \right| \tag{78}$$

$$S^*_{ab} = \int \left| \phi_a(\mathbf{r}) \, \phi_b(\mathbf{r}) \right| d\mathbf{r} \tag{79}$$

which is significantly stronger than the familiar [94–97] Schwarz bound (which is the first of the three braced quantities in (76)). Our next task is to compute shell-pair generalizations of (77)–(79) for each of the significant shell-pairs by evaluating (77)–(79) for each of their constituent basis-function-pairs and taking appropriate maxima [97].

The total computational effort involved in setting up the shell-pair data increases linearly with the size N of the basis. For tasks such as large Direct SCF calculations [44, 45], it is entirely negligible compared with the subsequent work; for less computationally demanding tasks, such as finding potential-derived atomic charges [98], it typically constitutes 10% of the job time.

4.2 Selection of Shell-Quartets

Given a sorted list of significant shell-pairs, we can construct all potentially important shell-quartets [99] by pairing the shell-pairs with one another. For the sake of vectorization, we deal with *batches* of shell-quartets of the same "type" and we utilize the memory which is available as effectively as possible in order to maximize the sizes of such batches [100].

Not every pair of shell-pairs, however, is necessarily accepted as a worthwhile shell-quartet. Although the shell-pair database has been carefully screened and contains no negligible shell-pairs, there are several ways in which a pair of significant shell-pairs may yield a shell-quartet which can be neglected...

(1) The quartet may be equivalent, by point group symmetry, to another quartet which has already been treated.

(2) The largest braket associated with the quartet may be negligibly small. This can be anticipated by the upper-bound formula (76) if the cutoff parameters (77)–(79) have been precomputed as indicated in Section 4.1.

(3) The largest density matrix (or delta-density matrix) elements which will multiply any of the brakets associated with the quartet may be negligibly small [44,97]. This is particularly common in late SCF cycles when incremental Fock matrix formation is being used.

Unfortunately, in general, it is not easy to vectorize the shell-quartet selection process because of the conditional nature of quartet acceptance. In the special case where one desires the electrostatic potential on a large grid, Johnson *et al.* have circumvented this problem by the so-called "fixed shell-pair" scheme [98] which is completely vectorizable. However, this approach is not directly applicable to the general case.

4.3 Generation of the $[0]^{(m)}$ Integrals

Given a batch of shell-quartets, the real computational work (denoted T_0 in Figures 1 and 2) can begin. In the first stage of this, the seven basic shell-quartet parameters

$$\mathbf{R} = \mathbf{Q} - \mathbf{P} \tag{80}$$

$$R^2 = R_x^2 + R_y^2 + R_z^2 \tag{81}$$

$$2\vartheta^2 = 1 \Big/ \left[\frac{1}{2\zeta} + \frac{1}{2\eta}\right] \tag{82}$$

$$2T = 2\vartheta^2 R^2 \tag{83}$$

$$U = U_P U_Q \tag{84}$$

are constructed. Given the shell-pair data which were generated earlier (Section 4.1), (80)–(84) can be computed in just 12 Flops and 17 Mops and this accounts for a rather small fraction of the total CPU time in most Direct SCF calculations (less than 4% in the pentacene run described in Section 4.7).

In the second stage, the $[\mathbf{0}]^{(m)}$ integrals ($0 \le m \le L$) are evaluated. If we desire two-electron repulsion integrals, *i.e.* $f(x) \equiv 1/x$ in (6), the relevant definition (Section 3.2.2) is

$$[\mathbf{0}]^{(m)} = U\left(2\vartheta^2\right)^{m+1/2} G_m(T) \tag{85}$$

where

$$G_m(T) = (2/\pi)^{1/2} \int_0^1 t^{2m} \exp\left(-Tt^2\right) dt \tag{86}$$

For other types of integrals, (85) and (86) must be appropriately modified but retain the same general form.

If T (which measures the extent to which the bra and ket charge distributions overlap) is less than a critical value T_{crit}, $G_L(T)$ can be evaluated using an interpolation procedure. We follow Shipman and Christoffersen [68] in favoring Chebyshev interpolation as an effective means for computing such functions but we have adopted the approach of Elbert and Davidson [69] who prefer to employ approximations of lower degree. We have chosen cubic interpolation as our standard and have discussed our methodology in detail in [71]. Values of $G_m(T)$ for $0 \le m < L$ can then be obtained, with numerical stability, by downward recursion (see (65), for example). The function exp(–T) is needed for this and, for speed, we also compute this by interpolation [71].

On the other hand, if T is greater than T_{crit}, the distributions overlap negligibly and (in the case of two-electron repulsion integrals) the $[\mathbf{0}]^{(m)}$ then reduce [101] to classical multipole terms

$$[\mathbf{0}]^{(m)} = \left(\frac{U}{R}\right)\left(\frac{1}{R^2}\right)\left(\frac{3}{R^2}\right)\cdots\left(\frac{2m-1}{R^2}\right) \qquad (87)$$

which can be computed recursively with great efficiency.

The subroutine in Gaussian 92 which is responsible for the evaluation of the $[\mathbf{0}]^{(m)}$ in the context of Direct SCF is Calc0m. This routine accounts for a noticeable fraction of the total CPU time (7% in the pentacene run described in Section 4.7) and has been carefully optimized. It runs at roughly 160 MFlops on a single-processor Cray Y-MP (whose theoretical peak speed is 333 MFlops).

In computations of the electrostatic potential on a grid, Gaussian 92 calls the subroutine Calc0G to compute the $[\mathbf{0}]^{(m)}$ and this routine accounts for roughly 50% of the total CPU time. As a result of the use of the "fixed shell pair" scheme [98] and very careful optimization, Calc0G runs at approximately 180 MFlops on the Cray Y-MP.

Given a batch of $[\mathbf{0}]^{(m)}$ integrals, it "only" remains to traverse one of the prisms in order to obtain the brakets which we seek. As mentioned earlier, this involves some combination of two Contraction steps and three Transformation steps and we now focus on these in detail.

4.4 Contraction Steps

The horizontal arrows in Figures 1 and 2 correspond to contraction steps and, in typical Direct SCF calculations using PRISM, these account for a significant fraction of the total CPU time (15% in the pentacene run described in Section 4.7). In electrostatic grid calculations, the fraction is even higher. It is therefore very important that they be executed as efficiently as possible.

As was indicated in Section 3.6, contraction within the braket framework involves the summation of primitive quantities which may, or may not, also need to be scaled before being added together. In the case where scaling is required, the contraction amounts to a dot product, *i.e.*

$$A_i = \sum_{j=1}^{K} S_{ij} B_{ij} \tag{88}$$

where i loops over the members of the batch and j loops over the length of the contraction. In Gaussian 92, (88) is implemented with i as the innermost loop and with the outer j loop unrolled sixfold. The resulting code, a representative kernel of which is

```
      DO 20 J=Jbeg,Jend,6
         DO 10 I=1,N
            A(I) = A(I) + S(I,J  )*B(I,J  ) + S(I,J+1)*B(I,J+1)
     $                  + S(I,J+2)*B(I,J+2) + S(I,J+3)*B(I,J+3)
     $                  + S(I,J+4)*B(I,J+4) + S(I,J+5)*B(I,J+5)
   10    CONTINUE
   20 CONTINUE
```

has a Flop/Mop ratio which approaches unity, possesses a good balance between adds and multiplies, is manifestly vectorizable and, not surprisingly, runs very fast on most platforms.

We note, too, that things only improve under the "fixed shell-pair" scheme which is used in electrostatic grid calculations because the scalings become loop-invariants [98].

4.5 Transformation Steps

As we indicated in Section 3.3, modern integral algorithms invariably employ recurrence relations to build complicated brakets from simple ones. Their use permits algorithms to deal (in principle) with brakets of arbitrarily high angular momentum and, additionally, to make good use of the intermediates that are shared by fraternal brakets. In the next four subsections, we will discuss the various recurrence relations which are used to move vertically on the prisms.

4.5.1 Two-Electron Transformations on the MD PRISM

The T_1^{MD} step in Figure 1 transforms $[\mathbf{0}]^{(m)}$ to $[\mathbf{r}]$ integrals and the T_2^{MD} and T_3^{MD} steps are the half-contracted and contracted variants of this. We will examine T_1^{MD} in detail because, although it is the simplest transformation step on either prism, it nonetheless shares many features with the more complicated transformation steps and, therefore, has useful pedagogical value.

The recurrence relation on which T_1^{MD}, T_2^{MD} and T_3^{MD} are based is the one-center RR (62)

$$[\mathbf{r}]^{(m)} \equiv R_i\,[\mathbf{r}-\mathbf{1}_i]^{(m+1)} - (r_i - 1)\,[\mathbf{r}-\mathbf{2}_i]^{(m+1)} \tag{89}$$

which McMurchie and Davidson derived originally using the elementary properties of Hermite polynomials. It is easily shown that, if the total angular momentum of the desired braket class is L, there are (L+1) $[\mathbf{0}]^{(m)}$ and (L+3)!/L!/3! $[\mathbf{r}]$ integrals and that, in general, the efficient generation of the latter from the former involves a complicated tree-search problem. Johnson *et al.* have carefully analysed this and have constructed highly optimized solution-trees [102]. In Gaussian 92, the subroutine MakMD1 employs these solutions in forming a "driver" for the T_1^{MD} step.

In Gaussian 92, a "driver" is an array of instructions (coded as integers) for the formation of one set of integrals from another. Subroutine MakMD1 first constructs a driver (the MD1 driver) to form the $[\mathbf{r}]$ from the $[\mathbf{0}]^{(m)}$ using (89). Given this driver, a set of $[\mathbf{0}]^{(m)}$ and the corresponding set of $\mathbf{R}$ vectors (80), subroutine DoMD1 then takes responsibility for the actual construction of the $[\mathbf{r}]$. For the purposes of illustration, it is useful to work through a simple example, such as L = 2.

The explicit L = 2 solution-tree for (89) is [102]

$$
\left.
\begin{aligned}
[s]^{(1)} &= [\mathbf{0}]^{(1)} \\
[p_z]^{(1)} &= R_z[\mathbf{0}]^{(2)} \\
[p_y]^{(1)} &= R_y[\mathbf{0}]^{(2)} \\
[p_x]^{(1)} &= R_x[\mathbf{0}]^{(2)} \\
[s] &= [\mathbf{0}]^{(0)} \\
[p_z] &= R_z[\mathbf{0}]^{(1)} \\
[p_y] &= R_y[\mathbf{0}]^{(1)} \\
[p_x] &= R_x[\mathbf{0}]^{(1)} \\
[d_{zz}] &= R_z[p_z]^{(1)} - [s]^{(1)} \\
[d_{yz}] &= R_y[p_z]^{(1)} \\
[d_{yy}] &= R_y[p_y]^{(1)} - [s]^{(1)} \\
[d_{xz}] &= R_z[p_x]^{(1)} \\
[d_{xy}] &= R_x[p_y]^{(1)}
\end{aligned}
\right\} \qquad (90)
$$

Thus, from $[\mathbf{0}]^{(0)}$, $[\mathbf{0}]^{(1)}$, $[\mathbf{0}]^{(2)}$, we form ten $[\mathbf{r}]$ integrals. We will assume (correctly) that the former are stored in locations 1, 2 and 3, respectively. Subroutines such as DoMD1 are very simple, containing only a handful of basic DO-loops corresponding to all useful special cases of the RR (89). The driver is a two-dimensional integer array, each row of which (an *instruction*) specifies the DO-loop and the locations to use. The MD1 driver for L = 2 is shown in Table III and DoMD1 is reproduced immediately below this.

Table III: The MD1 driver for L = 2				
4	3	0	1	0
5	3	0	2	0
3	3	0	3	0
6	4	2	1	1
7	5	0	1	0
4	4	0	3	0
5	5	2	2	1
8	3	0	2	0
3	3	2	3	1
9	2	0	1	0
10	2	0	2	0
2	2	0	3	0

```
      SUBROUTINE DoMD1(W,R,n,Driver,nDrive)
      IMPLICIT INTEGER (a-z)
      INTEGER Driver(5,nDrive)
      REAL*8 W(N,*),R(n,3)
      DO 40 knt = 1,nDrive
         Z     = Driver(1,knt)
         P1    = Driver(2,knt)
         P2    = Driver(3,knt)
         Axis  = Driver(4,knt)
         j     = Driver(5,knt)
         IF (j.eq.0) THEN
            DO 10 i = 1,n
               W(i,Z) = R(i,Axis) * W(i,P1)
10          CONTINUE
         ELSE IF (j.eq.1) THEN
            DO 20 i = 1,n
               W(i,Z) = R(i,Axis) * W(i,P1) - W(i,P2)
20          CONTINUE
         ELSE
            DO 30 i = 1,n
               W(i,Z) = R(i,Axis) * W(i,P1) - W(i,P2) * j
30          CONTINUE
         END IF
40    CONTINUE
      RETURN
      END
```

Each instruction in the driver is encoded by five integers:

(1) The location in which to store the result;

(2) The location of the first term in (89);

(3) The location of the second term in (89);

(4) The value of i in (89) ($1 \Rightarrow x$, $2 \Rightarrow y$, $3 \Rightarrow z$);

(5) The coefficient of the second term in (89).

For example, the first instruction (4, 3, 0, 1, 0) corresponds to the fourth equation in (90) and the fourth instruction (6, 4, 2, 1, 1) corresponds to the last equation in (90). We note the following:

(a) DoMD1 uses the coefficient of the second term in (89) to determine which of the three special cases of (89) to use.

(b) Each instruction results in the appropriate RR being applied to an entire *column* (n elements) of the W array, not just to a single element. That is, DoMD1 actually forms n sets of [**r**], not just a single set. Of course, n is the batch size described in Section 4.2 and this is the device by which PRISM is vectorized.

(c) The driver is frugal in its use of locations: to the greatest extent possible, it re-uses locations by *overwriting* intermediates when they are no longer needed. For example, instruction #1 places $[p_x]^{(1)}$ in location 4 but, after this has been used in instruction #4 to generate $[d_{xx}]$, the $[p_x]^{(1)}$ is overwritten in instruction #6 by $[d_{xz}]$.

The other PRISM subroutines whose names begin with Do (see Section 4.6) operate very similarly to DoMD1. Each is handed the W array, a driver and, if necessary, some auxiliary arrays such as R in DoMD1 and then proceeds to generate columns of W by combining columns with one another and, possibly, with the auxiliary arrays.

To complete this subsection, we must derive two further RRs. In order to undertake the T_2^{MD} or T_3^{MD} steps, we require RRs involving half-contracted or fully-contracted $[\mathbf{r}]^{(m)}$, respectively. These are easily derived [61c] by replacing R_i in (89) by appropriate identities. Thus, if the easily verified identity

$$R_i \equiv \frac{2\alpha}{2\zeta}(B_i - A_i) + (Q_i - B_i) \tag{91}$$

is substituted into (89), we obtain the half-contracted RR

$$\begin{aligned} {}_{a'b'p'}[\mathbf{r}]^{(m)} &= (B_i - A_i)_{(a'+1)b'(p'+1)}[\mathbf{r} - \mathbf{1}_i]^{(m+1)} \\ &+ (Q_i - B_i)_{a'b'p'}[\mathbf{r} - \mathbf{1}_i]^{(m+1)} \\ &- (r_i - 1)_{a'b'p'}[\mathbf{r} - \mathbf{2}_i]^{(m+1)} \end{aligned} \tag{92}$$

which is suitable for use in the T_2^{MD} step. Likewise, if the identity

$$R_i \equiv \frac{2\alpha}{2\zeta}(B_i - A_i) + \frac{2\gamma}{2\eta}(C_i - D_i) + (D_i - B_i) \tag{93}$$

is substituted into (89), we obtain the fully-contracted RR

$$\begin{aligned} {}_{a'b'p'}[\mathbf{r}]^{(m)}_{c'd'q'} &= (B_i - A_i)_{(a'+1)b'(p'+1)}[\mathbf{r} - \mathbf{1}_i]^{(m+1)}_{c'd'q'} \\ &+ (C_i - D_i)_{a'b'p'}[\mathbf{r} - \mathbf{1}_i]^{(m+1)}_{(c'+1)d'(q'+1)} \\ &+ (D_i - B_i)_{a'b'p'}[\mathbf{r} - \mathbf{1}_i]^{(m+1)}_{c'd'q'} \\ &- (r_i - 1)_{a'b'p'}[\mathbf{r} - \mathbf{2}_i]^{(m+1)}_{c'd'q'} \end{aligned} \tag{94}$$

which is suitable for use in the T_3^{MD} step.

4.5.2 One-Electron Transformations on the MD PRISM

The T_4^{MD}, T_7^{MD}, and T_8^{MD} steps in Figure 1 transform *uncontracted* p-bras (or q-kets) to *uncontracted* bras (or kets). There is little to say about such transformations except that they are accomplished by (57) and (59) and form the backbone of the classical MD algorithm [52]. Their chief weakness is that, because they are uncontracted transformations, they contribute to the x and y cost-parameters in (42) and therefore become very expensive when applied to braket classes of moderate or high degree of contraction. For such classes, it would obviously be more efficient to contract *before* transforming: this is precisely what the T_5^{MD}, T_6^{MD} and T_9^{MD} steps achieve. But how can this be done?

It is only the second terms in (57) and (59) which prevent the application of these RRs to contracted bras and kets because the prefactors $(P_i\text{-}A_i)$ and $(Q_i\text{-}C_i)$ vary from primitive to primitive. However, if we replace these by identities based on the definitions (12) of **P** and **Q**, as was first suggested in [61a], a new RR, which can be applied to *contracted* bras and kets, emerges. In bra notation, it can be expressed [61b] as

$$\left(\begin{matrix}\mathbf{0} & \mathbf{0} \\ \mathbf{a}+\mathbf{1}_i & \mathbf{b}\ \ \mathbf{p} \\ \mathbf{a}' & \mathbf{b}'\ \mathbf{p}'\end{matrix}\right| = P_i\left(\begin{matrix}\mathbf{0}\ \mathbf{0} \\ \mathbf{a}\ \ \mathbf{b}\ \ \mathbf{p}-\mathbf{1}_i \\ \mathbf{a}'\ \mathbf{b}'\ \ \mathbf{p}'\end{matrix}\right| + (B_i - A_i)\left(\begin{matrix}\mathbf{0} & \mathbf{0} \\ \mathbf{a} & \mathbf{b}\ \ \mathbf{p} \\ \mathbf{a}'\ \mathbf{b}'+\mathbf{1} & \mathbf{p}'+\mathbf{1}\end{matrix}\right| + \left(\begin{matrix}\mathbf{0}\ \mathbf{0} \\ \mathbf{a}\ \ \mathbf{b}\ \ \mathbf{p}+\mathbf{1}_i \\ \mathbf{a}'\ \mathbf{b}'\ \mathbf{p}'+\mathbf{1}\end{matrix}\right| \tag{95}$$

Eqn. (95) is the contracted analogue of (59) and is much more efficient for highly contracted braket classes. The corresponding contracted analogue of (57) is obvious.

Thus, by using (59), (89), (92), (94) and (95) in judicious combinations, one can traverse any of the 10 paths in Figure 1 (and, indeed, with only trivial extensions, any of the 20 paths in Figure 3). By choosing always to go by the cheapest (in a Flops or Mops sense) path, one gains the full benefit of the PRISM algorithm.

4.5.3 Two-Electron Transformations on the HGP PRISM

The T_1^{HGP} step in Figure 2 transforms $[\mathbf{0}]^{(m)}$ to $[\mathbf{m0|n0}]$ integrals and the T_2^{HGP} and T_3^{HGP} steps are the half-contracted and contracted variants of this. However, before we can even contemplate the HGP algorithm within a braket framework, it is necessary to recast the OS RR (67), which involves $\langle \mathbf{ab|cd} \rangle^{(m)}$ integrals, into a new form involving $[\mathbf{ab|cd}]^{(m)}$ integrals. Although this is straightforward to do, it has not previously appeared in the literature. Upon substituting the connection formula

$$[\mathbf{ab|cd}]^{(m)} \equiv \left(\frac{2\zeta\eta}{\zeta+\eta}\right)^m \langle \mathbf{ab|cd} \rangle^{(m)} \tag{96}$$

which is easily derived by comparing (63) and (68), into (67) and simplifying, we discover a new RR

$$\begin{aligned}
[(\mathbf{a}+\mathbf{1}_i)\mathbf{b|cd}]^{(m)} &= (B_i - A_i)\left(\frac{2\beta}{2\zeta}\right)[\mathbf{ab|cd}]^{(m)} + R_i\left(\frac{1}{2\zeta}\right)[\mathbf{ab|cd}]^{(m+1)} \\
&+ a_i\left\{\left(\frac{1}{2\zeta}\right)[(\mathbf{a}-\mathbf{1}_i)\mathbf{b|cd}]^{(m)} - \left(\frac{1}{2\zeta}\right)^2[(\mathbf{a}-\mathbf{1}_i)\mathbf{b|cd}]^{(m+1)}\right\} \\
&+ b_i\left\{\left(\frac{1}{2\zeta}\right)[\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}]^{(m)} - \left(\frac{1}{2\zeta}\right)^2[\mathbf{a}(\mathbf{b}-\mathbf{1}_i)|\mathbf{cd}]^{(m+1)}\right\} \\
&+ c_i\left(\frac{1}{2\zeta}\right)\left(\frac{1}{2\eta}\right)[\mathbf{ab}|(\mathbf{c}-\mathbf{1}_i)\mathbf{d}]^{(m+1)} + d_i\left(\frac{1}{2\zeta}\right)\left(\frac{1}{2\eta}\right)[\mathbf{ab}|\mathbf{c}(\mathbf{d}-\mathbf{1}_i)]^{(m+1)}
\end{aligned} \tag{97}$$

by which any $[\mathbf{ab|cd}]$ can be reduced to the $[\mathbf{00|00}]^{(m)} \equiv [\mathbf{0}]^{(m)}$ discussed in Section 4.3. Through this sleight of hand, we free ourselves of the necessity to consider the OS $\langle \mathbf{ab|cd} \rangle^{(m)}$ integrals any further and we greatly clarify the relationship between the MD and HGP algorithms. We note too that, in additon to being more aesthetically pleasing than (67), (97) is also computationally superior to it because all of the *four*-center exponent factors in (67) are replaced by *two*-center factors in (97). Thus, each of the terms in (97) is now clearly a braket, *i.e.* of the form (55).

In the original HGP method (Section 3.9), a special case of the OS RR, termed the VRR, is used to compute $[\mathbf{m}\,\mathbf{0}|\mathbf{n}\mathbf{0}]$ from $\langle\mathbf{00}|\mathbf{00}\rangle^{(\mathrm{m})}$ integrals. If, instead, we use the analogous special case of (97), we obtain a new RR

$$\begin{aligned}\left[(\mathbf{a}+\mathbf{1}_i)\mathbf{0}|\mathbf{c}\mathbf{0}\right]^{(\mathrm{m})} &= (B_i - A_i)\left(\frac{2\beta}{2\zeta}\right)\left[\mathbf{a}\mathbf{0}|\mathbf{c}\mathbf{0}\right]^{(\mathrm{m})} + R_i\left(\frac{1}{2\zeta}\right)\left[\mathbf{a}\mathbf{0}|\mathbf{c}\mathbf{0}\right]^{(\mathrm{m}+1)} \\ &+ a_i\left\{\left(\frac{1}{2\zeta}\right)\left[(\mathbf{a}-\mathbf{1}_i)\mathbf{0}|\mathbf{c}\mathbf{0}\right]^{(\mathrm{m})} - \left(\frac{1}{2\zeta}\right)^2\left[(\mathbf{a}-\mathbf{1}_i)\mathbf{0}|\mathbf{c}\mathbf{0}\right]^{(\mathrm{m}+1)}\right\} \\ &+ c_i\left(\frac{1}{2\zeta}\right)\left(\frac{1}{2\eta}\right)\left[\mathbf{a}\mathbf{0}|(\mathbf{c}-\mathbf{1}_i)\mathbf{0}\right]^{(\mathrm{m}+1)}\end{aligned} \tag{98}$$

by which we can compute $[\mathbf{m}\mathbf{0}|\mathbf{n}\mathbf{0}]$ from $[\mathbf{00}|\mathbf{00}]^{(\mathrm{m})} \equiv [\mathbf{0}]^{(\mathrm{m})}$. This, of course, is precisely what is needed to perform the T_1^{HGP} step in Figure 2.

Finally, just as we derived RRs for T_2^{MD} and T_3^{MD} from the RR for T_1^{MD} in Section 4.5.1, we can derive RRs for T_2^{HGP} and T_3^{HGP} from (98) by replacing R_i by the identities (91) and (93). In this way, we eventually obtain

$$\left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a}+\mathbf{1}_i & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a' & b' & p' & c' & d' & q' \end{array}\right)^{(m)} =$$

$$(B_i - A_i)\left\{\left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a' & b'+1 & p'+1 & c' & d' & q' \end{array}\right)^{(m)} + \left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a'+1 & b' & p'+2 & c' & d' & q' \end{array}\right)^{(m+1)}\right\}$$

$$+(C_i - D_i)\left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a' & b' & p'+1 & c'+1 & d' & q'+1 \end{array}\right)^{(m+1)}$$

$$+(D_i - B_i)\left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a' & b' & p'+1 & c' & d' & q' \end{array}\right)^{(m+1)}$$

$$+a_i\left\{\left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a}-\mathbf{1}_i & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a' & b' & p'+1 & c' & d' & q' \end{array}\right)^{(m)} - \left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a}-\mathbf{1}_i & \mathbf{b} & \mathbf{0} & \mathbf{c} & \mathbf{d} & \mathbf{0} \\ a' & b' & p'+2 & c' & d' & q' \end{array}\right)^{(m+1)}\right\}$$

$$+c_i\left(\begin{array}{ccc|ccc} \mathbf{0} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & \\ \mathbf{a} & \mathbf{b} & \mathbf{0} & \mathbf{c}-\mathbf{1}_i & \mathbf{d} & \mathbf{0} \\ a' & b' & p'+1 & c' & d' & q'+1 \end{array}\right)^{(m+1)} \tag{99}$$

for T_3^{HGP}. The derivation of the (slightly simpler) T_2^{HGP} recurrence relation is left as an exercise for the reader...

4.5.4 One-Electron Transformations on the HGP PRISM

The T_4^{HGP}, T_5^{HGP}, T_6^{HGP}, T_7^{HGP}, T_8^{HGP} and T_9^{HGP} steps in Figure 2 correspond to the transformation of [**m0**| and (**m0**| to [bra| and (bra|, respectively, and the analogous ket transformations. As Head-Gordon and Pople emphasized [55], the RR (70) which achieves this for (bra| = (**ab**| in the uncontracted case is also applicable in the contracted case. This represents an important difference between the MD and HGP PRISMs because it is therefore *always* preferable to contract fully before the last two transformations on the HGP PRISM whereas this is not the case on the MD PRISM.

At first glance, there is not much more that can be said about this transformation. The RR (70) is extremely simple and is easy to use and it might appear that our analysis can probe no further. However, as Ryu, Lee and Lindh have shown [59b], if one wishes to apply (70) in a way that minimizes the number of Flops involved, a complicated tree-search problem must first be solved. These authors were unable to solve the general problem but gave heuristic solutions which clearly indicated that substantial savings were available. However, this is not the approach which is followed in the HGP-PRISM algorithm...

As Johnson *et al.* have recently found [63], if one seeks a transformation scheme which is *Mop-optimal*, rather than Flop-optimal, one is led to introduce an entire new *family* of RRs, which these authors terms "nth-order transfer relations". The 1st-order transfer relation is simply (70); the 2nd-order transfer relations are obtained by applying (70) to itself; and so on. The interested reader is referred to the original literature for the explicit forms of the first few transfer relations.

One is immediately led to ask why the Flop-optimal and Mop-optimal solutions should be so different and the answer, surprisingly, lies in the extreme simplicity of (70). Although it contains two Flops, it involves four Mops and this is an unhealthy balance: in essence, not enough "real work" is done between loading the right-hand-side and saving the left-hand-side. Instead, by the use of comparatively long nth-order transfer relations, we reduce the amount of memory traffic dramatically and, on many modern machines, the CPU cost falls. This is particularly revealing because the Mop-optimal solutions frequently boast Flop costs of disconcertingly large proportions.

4.6 Loop Structure of PRISM in Gaussian 92

Call **ListS2** to form list of significant shell-pairs
Call **SortS2** to sort shell-pair data by type
Call **CutoS2** to compute cutoff parameters for each shell-pair
Loop over LTot values
  Call **TabGmT** to set up appropriate interpolation tables
  Call **MakMD1** to make $[\mathbf{0}]^{(m)} \rightarrow [\mathbf{r}]$ driver
  Loop over bra angular momentum types
    Call several routines to form bra-transformation drivers
    Loop over ket angular momentum types
      Call several routines to form ket-transformation drivers
      Call **FillAv** to decide which paths to make available
      Call **MkDrTp** to make transposition drivers
      Call **MakVR1** to make $[\mathbf{0}]^{(m)} \rightarrow [\mathbf{m0|n0}]$ driver
      Call **MkDrPQ** to make scatter drivers
      Call several routines to make $(\mathbf{0})^{(m)} \rightarrow (\mathbf{r})$ drivers
      Call several routines to make contraction drivers
      Call **MkCost** to compute PRISM step-costs in Flops and Mops
      Loop over KBra values
        Form petite list of bra shell-pairs of current type
        Loop over KKet values
          Call **Choose** to select the cheapest path
          Call **PthInf** to determine info about the chosen path
          Call **CalcSF** to compute bra- and ket-scalings
          Compute maximum number of quartets per batch
          Loop over batches of quartets of current type
            Call **PickS4** to select a batch of quartets
            Call **LoadS4** to form scaling and distance arrays
            Call **CalcS4** to compute R2, T, Theta and U values
            Call **Calc0m** to compute $[\mathbf{0}]^{(m)}$ integrals
            Call **DoMD1** to transform $[\mathbf{0}]^{(m)} \rightarrow [\mathbf{r}]$
            Call **DoCont** to contract $[\mathbf{r}] \rightarrow (\mathbf{r})$
            Call **DoCont** to contract $(\mathbf{r}] \rightarrow (\mathbf{r})$
            Call **DoShuf** to scatter $(\mathbf{r}) \rightarrow (\mathbf{p|q})$
            Call **DoTran** to transform $(\mathbf{p|q}) \rightarrow (\mathbf{bra|q})$
            Call **DoShuf** to transpose $(\mathbf{bra|q})$
            Call **DoTran** to transform $(\mathbf{bra|q}) \rightarrow (\mathbf{bra|ket})$
            Call **Gobbxx** to digest (bra|ket)'s
            Call **Scatxx** to scatter Fock contributions etc.

4.7 Performance of PRISM in Gaussian 92

Notwithstanding the apparently strong *theoretical* arguments in support of the utility of the PRISM philosophy, our efforts are in vain unless it can be demonstrated that a real computer program, employing the algorithm, runs fast. In their paper on the HGP algorithm [55], Head-Gordon and Pople presented CPU timings on various computers which suggested that the HGP method was significantly faster than any other method which existed at that time (1988), with thepossible exception of the PH axis-switch technique (Section 3.5). On that basis, we have systematically used the HGP method as a performance target while developing the PRISM methods and have published results [61] which establish clearly that the MD-PRISM algorithm is generally superior to the HGP algorithm. However, it was noted in the conclusions of [61c] that the MD-PRISM "is substantially *inferior* to HGP for weakly contracted classes of high angular momentum" and that "a modified version of PRISM that does not suffer from this defect needs to be developed". We believe that, by incorporating paths on the new *HGP* PRISM, we now have an algorithm which is *uniformly and significantly* superior to all existing methods.

As an indication of the performance of our implementation [103] of PRISM in Gaussian 92, we have measured the CPU time for a single Hartree-Fock SCF iteration on a number of polyacenes. Our aim in making these measurements is to establish some well-defined benchmarks against which other programs in the future can be tested.

The precise specifications underlying our timings are:

(1)	Computer:	Dedicated IBM RS/6000 Model 320 (AIX)
(2)	Procedure:	Direct RHF-SCF (1st cycle only)
(3)	Basis Set:	6–31G*
(4)	Accuracy:	10^{-10}
(5)	Symmetry Used:	D_2
(6)	Initial Guess:	Projected INDO
(7)	Geometry:	C-C bond lengths = 1.4 Å
		C-H bond lengths = 1.1 Å
		All angles = 120 degrees.

Table IV: Timings[a] for One Direct HF–SCF Cycle on Some Polyacenes

Molecule	N[b]	Integrals[c]	Fock[d]	Total[e]
C_6H_6	102	38 (61)	13	52 (74)
$C_{10}H_8$	166	163 (269)	67	230 (336)
$C_{14}H_{10}$	230	403 (652)	191	594 (843)
$C_{18}H_{12}$	294	747 (1215)	371	1118 (1585)
$C_{22}H_{14}$	358	1192 (1956)	627	1819 (2583)

(a) In CPU seconds with HGP values in parentheses.
See text for detailed specifications.
(b) Number of basis functions.
(c) Time to construct all needed integrals.
(d) Time to digest integrals into an RHF Fock matrix.
(e) Sum of (c) and (d).

We may infer from the data in Table IV that the cutoff scheme used in PRISM is working satisfactorily. Even though none of the molecules considered could be considered large, the CPU times are already increasing much less rapidly than N^4: between $C_{18}H_{12}$ and $C_{22}H_{14}$ the functional dependence is close to $N^{2.5}$.

It is also interesting to note that the fraction of the Total time which is associated with Fock matrix construction becomes larger in the larger systems. Presumably, this is because (due to our modeling scheme) the average degree of contraction is less in the larger molecules. This renders the integrals cheaper to form but has no effect on the Fock construction time.

5. PROSPECTS FOR THE FUTURE

The theory and practice of Molecular Integrals over Gaussian Basis Functions have come a long way since Boys' 1950 proposal. However, notwithstanding the enormous progress that has been made, there is little justification for the notion that we have reached the end of the road. Today, one can undertake Quantum Chemical calculations which would have been inconceivable only a decade ago and, without a doubt, computational chemists will be making the same observation a decade from now.

Advances in computer hardware will continue to catalyse and fuel the construction of novel software strategies. The advent of vector machines during the 1980s revolutionized the way in which integral programs were constructed and a second revolution is now underway in response to the possibilities afforded by the new Massively Parallel Processing (MPP) machines. The problem of effectively implementing PRISM within an MPP framework, which is presently under investigation in a number of groups, is a difficult one and it may eventually turn out that only radical departures from the conventional wisdom will yield efficient codes on the teraflop computers of the future.

Another, equally important, avenue to future developments involves the importation of ideas from other disciplines. This type of cross-fertilization – a ubiquitous ingredient in the advance of science – is apparent in the papers of Feibelman [104], Yang [105], Panas and Almlöf [106], Galli and Parrinello [107] and others. Indeed, some of these workers provide tantalizing evidence that the Holy Grail of Quantum Chemistry – a practical scheme for SCF calculations whose expense increases asymptotically only *linearly* with the system size – may be achieved in the rather near future.

Speculatively combining such hardware and software advances leads to the conclusion that accurate *ab initio* calculations on molecular systems with thousands of atoms may soon be routine and Quantum Chemistry will have come of age.

6. ACKNOWLEDGMENTS

It is a pleasure to acknowledge the contributions which many others have made to my research effort at Carnegie Mellon University.

For more than four years, John Pople has been friend and mentor to me. Like others before me, I have been greatly influenced by the seasoned pragmatism which underlies his approach to Quantum Chemistry.

In a similar vein, it was a number of conversations with Martin Head-Gordon, shortly after his discovery of the algorithm which now bears his name, that sparked my early interest in the theory of Molecular Integrals.

More recently, the careful work of Benny Johnson has been instrumental in elucidating how best to apply PRISM in the context of the two- and three-center problems. The extraordinary speed of the Electrostatic Potential program in Gaussian 92 is compelling testimony to his successes in this endeavor.

The greater part of what I know about constructing effective computer implementations of abstract algorithms I have learned from Mike Frisch and I am indebted to him for his advice over the years.

The MD-PRISM was first presented at the Argonne National Laboratory's "Methodology of the Evaluation of Integrals in LCAO Calculations" workshop in August 1990. The HGP-PRISM was first presented at the "PRISM" workshop in Connecticut in August 1992. I am grateful to Argonne and to Lorentzian Inc., respectively, for invitations to speak at these meetings.

My work at Carnegie Mellon has been funded under various National Science Foundation grants.

Finally, I thank my wife, Elizabeth, for her loving constancy, her encouragement and her typing of this manuscript.

7. REFERENCES

[1] E. Schrödinger, *Ann. Physik* 79 (1926) 361.

[2] (a) J.C. Slater, *Phys. Rev.* 34 (1929) 1293.
(b) J.C. Slater, *Phys. Rev.* 35 (1930) 509.

[3] (a) D.R. Hartree, *Proc. Cam. Phil. Soc.* 24 (1928) 89.
(b) D.R. Hartree, *Proc. Cam. Phil. Soc.* 24 (1928) 111.
(c) D.R. Hartree, *Proc. Cam. Phil. Soc.* 24 (1928) 426.

[4] V. Fock, Z. *Physik* 61 (1930) 126.

[5] P. Hohenberg and W. Kohn, *Phys. Rev. B*136 (1964) 864.

[6] W. Kohn and L.J. Sham, *Phys. Rev. A*140 (1965) 1133.

[7] L.H. Thomas, *Proc. Cam. Phil. Soc.* 23 (1927) 542.

[8] E. Fermi, *Rend. Accad., Lincei* 6 (1927) 602.

[9] P.A.M. Dirac, *Proc. Cam. Phil. Soc.* 26 (1930) 376.

[10] J.C. Slater, *Phys. Rev.* 81 (1951) 385.

[11] (a) R. Pariser and R.G. Parr, *J. Chem. Phys.* 21 (1953) 466.
(b) R. Pariser and R.G. Parr, *J. Chem. Phys.* 21 (1953) 767.
(c) J.A. Pople, *Trans. Faraday Soc.* 49 (1953) 1375.

[12] J.A. Pople and D.L. Beveridge, *Approximate Molecular Orbital Theory* (McGraw-Hill, New York, 1970)

[13] R.C. Bingham, M.J.S. Dewar and D.H. Lo, *J. Am. Chem. Soc.* 97 (1975) 1285.

[14] M.J.S. Dewar and W. Thiel, *J. Am. Chem. Soc.* 99 (1977) 4899.

[15] M.J.S. Dewar, E.G. Zoebisch, E.F. Healy and J.J.P. Stewart, *J. Am. Chem. Soc.* 107 (1985) 3902.

[16] J.J.P. Stewart, *J. Comput. Chem.* 10 (1989) 221.

[17] A. Szabo and N.S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Quantum Chemistry* (McGraw-Hill, New York, 1989).

[18] The Kohn-Sham equations would indeed be equivalent to the Schrödinger equation if the universal exchange-correlation functional were known. Since it has not yet been discovered, practical forms of Kohn-Sham theory are (like Hartree-Fock theory) well-defined approximations to Schrödinger theory.

[19] (a) P. Pulay, *Adv. Chem. Phys.* 69 (1987) 241.
(b) H.B. Schlegel, *Adv. Chem. Phys.* 67 (1987) 249.
(c) R. Fournier, J. Andzelm and D.R. Salahub, *J. Chem. Phys.* 90 (1989) 6371.

[20] (a) J.A. Pople, R. Krishnan, H.B. Schlegel and J.S. Binkley, *Int. J. Quantum Chem., Symp.* 13 (1979) 225.
(b) R. Fournier, *J. Chem. Phys.* 92 (1990) 5422.
[21] (a) S.M. Colwell, D. Jayatilaka, P.E. Maslen, R.D. Amos and N.C. Handy, *Int. J. Quantum. Chem.* 40 (1991) 179.
(b) P.E. Maslen, D. Jayatilaka, S.M. Colwell, R.D. Amos and N.C. Handy, *J. Chem. Phys.* 95 (1991) 7409.
(c) P.E. Maslen, N.C. Handy, R.D. Amos and D. Jayatilaka, *J. Chem. Phys.* 97 (1992) 4233.
[22] J.C. Slater, *Phys. Rev.* 36 (1930) 57.
[23] W.J. Hehre, R.F. Stewart and J.A. Pople, *J. Chem. Phys.* 51 (1969) 2657.
[24] H.J. Monkhorst and F.E. Harris, *Chem. Phys. Letters* 3 (1969) 537.
[25] S.F. Boys, *Proc. Roy. Soc. (London) A*200 (1950) 542.
[26] E.R. Davidson and D. Feller, *Chem. Rev.* 86 (1988) 681.
[27] E. Clementi, *IBM J. Res. and Dev.* 9 (1965) 2.
[28] (a) H. Preuss, *Z. Naturforsch.* 11a (1956) 823.
(b) J.L. Whitten, *J. Chem. Phys.* 39 (1963) 349.
(c) H. Sambe, *J. Chem. Phys.* 42 (1965) 1732.
(d) J.F. Harrison, *J. Chem. Phys.* 46 (1967) 1115.
(e) A.A. Frost, *J. Chem. Phys.* 47 (1967) 3707.
[29] (a) E.A. McCullough Jr., *Chem. Phys. Letters* 24 (1974) 55.
(b) E.A. McCullough Jr., *J. Chem. Phys.* 62 (1975) 3991.
(c) E.A. McCullough Jr., *Comp. Phys. Rep.* 4 (1986) 265.
[30] (a) L. Adamowicz and E.A. McCullough Jr., *J. Chem. Phys* 75 (1981) 2475.
(b) L. Adamowicz, R.J. Bartlett and E.A. McCullough Jr., *Phys. Rev. Lett.* 54 (1985) 426.
[31] (a) A.D. Becke, *J. Chem. Phys.* 76 (1982) 6037.
(b) A.D. Becke, *J. Chem. Phys.* 78 (1983) 4787.
(c) A.D. Becke, *Phys. Rev. A* 33 (1986) 2786.
[32] (a) L. Laaksonen, D. Sundholm and P. Pyykkö, *Int. J. Quantum Chem.* 27 (1985) 601.
(b) L. Laaksonen, P. Pyykkö and D. Sundholm, *Comp. Phys. Rep.* 4 (1986) 313.
[33] D. Heinemann, B. Fricke and D. Kolb, *Chem. Phys. Letters* 145 (1988) 125.
[34] M. Defranceschi, M. Suard and G. Berthier, *Int. J. Quantum Chem.* 25 (1984) 863.
[35] S.A. Alexander, R.L. Coldwell and H.J. Monkhorst, *J. Comp. Phys.* 76 (1988) 263.

[36] (a) A.D. Becke and R.M. Dickson, *J. Chem. Phys.* 89 (1988) 2993.
(b) A.D. Becke, *Int. J. Quantum Chem., Symp.* 23 (1989) 599.
(c) A.D. Becke, *J. Chem. Phys.* 92 (1990) 3610.
(d) A.D. Becke, *J. Chem. Phys.* 96 (1992) 2155.

[37] (a) V.I. Lebedev, *Zh. Vychisl. Mat. Mat. Fiz.* 15 (1975) 48.
(b) V.I. Lebedev, *Zh. Vychisl. Mat. Mat. Fiz.* 16 (1976) 293.
(c) V.I. Lebedev, *Sibirsk. Mat. Zh.* 18 (1977) 132.
(d) V.I. Lebedev, *Proc. Conf. Novosibirsk (1978)*, ed. S.L. Sobolev (Nauka Sibirsk. Otdel., Novosibirsk, 1980).

[38] A.D. Becke, *J. Chem. Phys.* 88 (1988) 2547.

[39] B. Delley, *J. Chem. Phys.* 92 (1990) 508.

[40] D.R. Salahub, in *Density Functional Methods in Chemistry*, eds. J.K. Labanowski and J.W. Andzelm (Springer-Verlag, New York, 1991).

[41] C.W. Murray, N.C. Handy and G.J. Laming, *Mol. Phys.* in press.

[42] P.M.W. Gill, B.G. Johnson and J.A. Pople, *Chem. Phys. Letters*, submitted.

[43] Overlap, kinetic-energy, nuclear-attraction and two-electron-repulsion integrals can all be written in this general form. The matrix elements of the exchange-correlation contribution to the energy in Kohn-Sham theory cannot, however. They are so awkward that a second basis set (usually of the Gaussian or Delta type) must be introduced to permit the computation. This, however, lies beyond the scope of our present Review.

[44] J. Almlöf, K. Faegri and K. Korsell, *J. Comput. Chem.* 3 (1982) 385.

[45] J. Andzelm, in *Density Functional Methods in Chemistry*, eds. J.K Labanowski and J.W. Andzelm (Springer-Verlag, New York, 1991).

[46] I. Shavitt, in *Methods in Computational Physics, Vol. 2,* eds. B. Alder, S. Fernback and M. Rotenberg (Academic Press, New York, 1963).

[47] H. Taketa, S. Huzinaga and K. O-ohata, *J. Phys. Soc. Japan* 21 (1966) 2313.

[48] E. Clementi and D.R. Davis, *J. Comput. Phys.* 1 (1967) 223.

[49] (a) E. Clementi, H. Clementi and D.R. Davis, *J. Chem. Phys.* 46 (1967) 4725.
(b) E. Clementi, *J. Chem. Phys.* 46 (1967) 4731.
(c) E. Clementi, *J. Chem. Phys.* 46 (1967) 4737.

[50] (a) J.A. Pople and W.J. Hehre, *J. Comp. Phys.* 27 (1978) 161.
(b) W.J. Hehre, W.A. Lathan, M.D. Newton, R. Ditchfield and J.A. Pople, Gaussian 70, Program No. 136, QCPE, Indiana University, Bloomingtion, Indiana.

[51] (a) M. Dupuis, J. Rys and H.F. King, *J. Chem. Phys.* 65 (1976) 111.
(b) H.F. King and M. Dupuis, *J. Comput. Phys.* 21 (1976) 44.
(c) J. Rys, M. Dupuis and H.F. King, *J. Comput. Chem.* 4 (1983) 154.
[52] L.E. McMurchie and E.R. Davidson, *J. Comput. Phys.* 26 (1978) 218.
[53] (a) H.B. Schlegel, *J. Chem. Phys.* 77 (1982) 3676.
(b) H.B. Schlegel, *J. Chem. Phys.* 90 (1989) 5630.
[54] (a) S. Obara and A. Saika, *J. Chem. Phys.* 84 (1986) 3963.
(b) S. Obara and A. Saika, *J. Chem. Phys.* 89 (1988) 1540.
[55] M. Head-Gordon and J.A. Pople, *J. Chem. Phys.* 89 (1988) 5777.
[56] K. Ishida, *J. Chem. Phys.* 95 (1991) 5198.
[57] T.P. Hamilton and H.F. Schafer III, *Chem. Phys.* 150 (1991) 163.
[58] I. Panas, *Chem. Phys. Letters* 184 (1991) 86.
[59] (a) R. Lindh, U. Ryu and B. Liu, *J. Chem. Phys.* 95 (1991) 5889.
(b) U. Ryu, Y.S. Lee and R. Lindh, *Chem. Phys. Letters* 185 (1991) 562.
[60] T. Helgaker and P.R. Taylor, *Theor. Chim. Acta* 83 (1992) 177.
[61] (a) P.M.W. Gill, M. Head-Gordon and J.A. Pople, *Int. J. Quantum Chem., Symp.* 23 (1989) 269.
(b) P.M.W. Gill, M. Head-Gordon and J.A. Pople, *J. Phys. Chem.* 94 (1990) 5564.
(c) P.M.W. Gill and J.A. Pople, *Int. J. Quantum Chem.* 40 (1991) 753.
[62] M.J. Frisch, B.G. Johnson, P.M.W. Gill, D.J. Fox and R.H. Nobes, *Chem. Phys. Letters,* submitted.
[63] B.G. Johnson, P.M.W. Gill and J.A. Pople, *Chem. Phys. Letters*, submitted.
[64] V.R. Saunders, in *Computational Techniques in Quantum Chemistry and Molecular Physics,* eds. G.H.F. Diercksen, B.T. Sutcliffe and A. Veillard (Reidel, Dordrecht, 1975).
[65] V.R. Saunders, in *Methods in Computational Physics,* eds. G.H.F. Diercksen, and S. Wilson (Reidel, Dordrecht, 1983).
[66] D. Hegarty and G. van der Velde, *Int. J. Quantum Chem.* 23 (1983) 1135.
[67] D. Hegarty, in *Advanced Theories and Computational Approaches to the Electronic Structure of Molecules,* ed. C.E. Dykstra (Reidel, Dordrecht, 1984).
[68] L.L. Shipman and R.E. Christoffersen, *Comp. Phys. Comm.* 2 (1971) 201.
[69] S.T. Elbert and E.R. Davidson, *J. Comp. Phys.* 16 (1974) 391.
[70] F.E. Harris, *Int. J. Quantum Chem.* 23 (1983) 1469.
[71] P.M.W. Gill, B.G. Johnson and J.A. Pople, *Int. J. Quantum Chem.* 40 (1991) 745.

[72] (a) P.M.W. Gill, B.G. Johnson, J.A. Pople and S.W. Taylor, *J. Chem. Phys.* 96 (1992) 7178.
(b) P.M.W. Gill and J.A. Pople, in preparation.
[73] (a) W. Kutzelnigg, *Theor. Chim. Acta* 68 (1985) 445.
(b) W. Kutzelnigg and W. Klopper, *J. Chem. Phys.* 94 (1991) 1985.
[74] A. Preiskorn and B. Zurawski, *Int. J. Quantum Chem.* 27 (1985) 641.
[75] M.J. Bearpart, N.C. Handy, R.D. Amos and P.E. Maslen, *Theor. Chim. Acta* 79 (1991) 361.
[76] W. Klopper and R. Röhse, *Theor. Chim. Acta* 83 (1992) 441.
[77] J.S. Binkley, M.J. Frisch, D. J. Defrees, K. Raghavachari, R.A. Whiteside, H.B. Schlegel, E.M. Fluder and J.A. Pople, Gaussian 82, Carnegie-Mellon University, Pittsburgh (1984).
[78] M.J. Frisch, J.S. Binkley, H.B. Schlegel, K. Raghavachari, C.F. Melius, R.L. Martin, J.J.P. Stewart, F.W. Bobrowicz, R.A. Whiteside, D.J. Fox, E.M.Fleuder and J.A. Pople, Gaussian 86, Carnegie Mellon Quantum Chemisty Publishing Unit, Pittsburgh, PA (1987).
[79] One axis coincides with $\overline{\mathbf{AB}}$ another passes through **Q**, and the third is their mutual perpendicular.
[80] One axis coincides with $\overline{\mathbf{AB}}$, another is mutually perpendicular to $\overline{\mathbf{AB}}$ and $\overline{\mathbf{CD}}$, and the third is their mutual perpendicular.
[81] P.C. Hariharan, PhD thesis, Carnegie Mellon University (1973).
[82] Hamilton and Schaefer [57] ascribe this idea to C. Murray.
[83] A. Komornicki, K. Ishida, M. Morokuma, R. Ditchfield and M. Conrad, *Chem. Phys. Letters* 45 (1977) 595.
[84] Gaussian 92, 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 Inc., Pittsburgh PA, 1992.
[85] Pairs of shells are required for most PRISM calculations but, for example, if overlap integrals are being calculated as outlined in Section 3.2.1, the "shell-pairs" are constructed by pairing each shell with a "dummy" ($\beta = 0$) shell.
[86] D.L. Wilhite and R.N. Euwema, *J. Chem. Phys.* 61 (1974) 375.
[87] J. Rys, H.F. King and P. Coppens, *Chem. Phys. Letters* 41 (1976) 383.
[88] M. Yanez, R.F. Stewart and J.A. Pople, *Acta. Cryst. A* 34 (1978) 641.

[89] (a) B.I. Dunlap, J.W.D. Connolly and J.R. Sabin, *J. Chem. Phys.* 71 (1979) 3396.
(b) B.I. Dunlap, J.W.D. Connolly and J.R. Sabin, *J. Chem. Phys.* 71 (1979) 4993.

[90] (a) G.G. Hall and D. Martin, *Isr. J. Chem.* 19 (1980) 255.
(b) G.G. Hall, *Theor. Chim. Acta* 63 (1983) 357.
(c) G.G. Hall and C.M. Smith, *Int. J. Quantum Chem.* 25 (1984) 881.
(d) C.M. Smith and G.G. Hall, *Theor. Chim. Acta* 69 (1986) 63.

[91] C. Van Alsenoy, *J. Comput. Chem.* 9 (1988) 620.

[92] (a) A. Fortunelli and O. Salvetti, *J. Comput. Chem* 12 (1991) 36.
(b) A. Fortunelli and O. Salvetti, *Chem. Phys. Letters* 186 (1991) 372.

[93] P.M.W. Gill, B.G. Johnson and J.A. Pople, *Chem. Phys. Letters*, submitted.

[94] J.C. Maxwell, *A Treatise on Electricity and Magnetism* (Clarendon, London, 1904).

[95] J.L. Whitten, *J. Chem. Phys.* 58 (1973) 4496.

[96] J. Power and R.M. Pitzer, *Chem. Phys. Letters* 24 (1974) 478.

[97] (a) R. Ahlrichs, *Theor. Chim. Acta* 33 (1974) 157.
(b) M. Haser and R. Ahlrichs, *J. Comput. Chem.* 10 (1989) 104.
(c) H. Horn, H. Weiβ, M. Haser, M. Ehrig and R. Ahlrichs, *J. Comput. Chem.* 12 (1991) 1058.

[98] B.G. Johnson, P.M.W. Gill, J.A. Pople and D.J. Fox, *Chem. Phys. Letters,* submitted.

[99] Pairs of shell-pairs are required for most PRISM calculations but, for example, if nuclear-attraction integrals are being calculated as outlined in Section 3.2.4, the "shell-quartets" are constructed by pairing the shell-pairs with the infinite-exponent Gaussian at each of the nuclei.

[100] For example, we might begin by forming batches of (*ss*|*ss*) quartets with $K_{bra} = K_{ket} = 1$ and, once these have all been treated, we might proceed with batches of (*ss*|*ss*) quartets with $K_{bra} = 2$ and $K_{ket} = 1$, and so forth. This approach, which is known as *extrinsic* vectorization, has been discussed by a number of authors. Our version is described in detail in [61b].

[101] Eq. (87) is readily derived from (86) by examining the asymptotic expansion of the latter and it is the attractive simplicity of (87) which motivates the definition (74).

[102] B.G. Johnson, P.M.W. Gill and J.A. Pople, *Int. J. Quantum Chem.* 40 (1991) 809.

[103] The following paths were made available in these calculations: MD–CCTTT, MD–CTCTT, MD–TCCTT, MD–TCTCT, MD–TTTCC and HGP–TCCTT.
[104] P.J. Feibelman, *J. Chem. Phys.* 81 (1984) 5864.
[105] W. Yang, *Phys. Rev. Lett.* 66 (1991) 1438.
[106] I. Panas and J. Almlöf, *Int. J. Quantum Chem.* 42 (1992) 1073.
[107] G. Galli and M. Parrinello, *Phys. Rev. Lett.* 69 (1992) 3547.