3

Four-Index Transformations

S. WILSON

1. Introduction

Atomic and molecular electronic structure calculations are most frequently performed by employing basis set expansion techniques; that is, by invoking the algebraic approximation (for recent reviews see Refs. 1 and 2). In electronic structure calculations which go beyond an independent electron or orbital model and take account of electron correlation effects, it is necessary to perform, either explicitly or implicitly, a transformation. Specifically, it is necessary to carry out a transformation of integrals involving the components of the Schrödinger operator over the chosen basis functions, usually either exponential-type functions or Gaussian-type functions, to integrals over the orbitals resulting from the independent electron model calculation, usually a self-consistent-field calculation.

Computationally, the orbital transformation stage of an atomic or molecular electronic structure calculation is an insignificant problem for integrals involving one-electron operators, but for the integrals involving two-electron operators it can become the most demanding phase of the entire calculation. The amount of computation required to perform a four-index transformation depends on the fifth power of the number of basis functions. By way of comparison, it should be noted that integral evaluation increases as the fourth power of the number of basis functions as do self-consistent-field calculations and second-order many-body perturbation theory studies of correlation effects. It is, therefore, essential to have

S. WILSON • University of Manchester Regional Computer Centre, Manchester M13 9PL, England.

S. Wilson (ed.), *Methods in Computational Chemistry*

efficient algorithms for performing the four-index transformation of two-electron integrals in electronic structure calculations in which correlation effects are accounted for.

Because of its conceptual simplicity, the four-index transformation has, quite justifiably, been given only a fraction of the attention that has been commanded in the literature by the electron correlation problem. It is, however, an essential first step in any calculation that aims to account for correlation effects. In view of its computational demands, we devote this chapter of a volume devoted to the electron correlation problem to the task of devising efficient four-index transformation schemes for two-electron integrals.

The plan of this chapter is as follows: In Section 2 we discuss the two-electron integrals, paying particular attention to their permutation symmetry properties, classification, labelling, and storage. The four-index transformation is considered in detail in Section 3. We consider the methods of Bender,[3] and Shavitt,[4] Elbert,[5] and Saunders and van Lenthe.[6] In Section 4 we discuss the exploitation of point group symmetries in the four-index transformation. Applications to atoms, to diatomic and linear molecules, and to polyatomic molecules are considered. Partial four-index transformations are considered in Section 5. We consider the methods of Pendergast and Hayes[7] and of Saunders and van Lenthe.[6] Some techniques that have been proposed for making controlled approximations in the four-index transformation are discussed in Section 6. Particular emphasis is given to the use of the Cholesky decomposition of the two-electron integral matrix as suggested by Beebe and Linderberg.[8] Simplifications in the transformation stage of an electronic structure calculation that result from the use of universal even-tempered basis sets[1] are underlined. Methods that combine the four-index transformation and the electron correlation energy calculation are described in Section 7. We discuss the possibility of implementing a direct transformation scheme that would avoid the need to store two-electron integrals either over the basis functions or over the orbitals. In Section 8, we discuss the impact of vector processing computers on the four-index transformation, and in Section 9 the possible impact of parallel processing computers is considered. Finally, in Section 10, we provide an overview of the development of efficient four-index transformations to date and suggest some future directions for investigation.

2. Two-Electron Integrals

2.1. Electron–Electron Repulsion Integrals

The two-electron Coulomb repulsion integrals which arise in atomic and molecular electronic structure studies are defined as follows:

$$\int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_p^*(\mathbf{r}_1) \, \chi_q^*(\mathbf{r}_2) \, r_{12}^{-1} \chi_r(\mathbf{r}_1) \, \chi_s(\mathbf{r}_2) \tag{1}$$

in which the χ_p are one-electron basis functions and $r_{12} = |r_1 - r_2|$ is the interelectronic distance. In the bra–ket notation of Dirac this integral may be denoted by

$$\langle pq | \, r_{12}^{-1} \, | rs \rangle \tag{2}$$

or, more simply, as

$$\langle pq \, | \, rs \rangle \tag{3}$$

In an alternative notation, the charge cloud notation,[1] the two-electron integral is labelled by

$$[pr \, | \, qs] \tag{4}$$

in which the indices associated with the electron labelled 1 are placed to the left and those associated with the electron labelled 2 are placed to the right.

The evaluation of the two-electron Coulomb repulsion integrals forms a major component of the computational resources required to execute molecular electronic structure programs, particularly in the case of non-linear polyatomic molecules when basis sets of exponential-type functions are used. In the present chapter, however, we are not concerned with the evaluation of these integrals and we refer the reader interested in further details of this aspect of electronic structure calculations elsewhere.[9–11]

2.2. Permutational Symmetry

Because the interelectronic Coulomb repulsion operator is symmetric in the coordinates of the electrons, the two-electron integrals involving this component of the Schrödinger operator have certain permutational symmetry properties that can be exploited in all phases of electronic structure studies including the transformation.

For complex one-electron functions we have the equalities

$$\begin{aligned}
&\int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_p^*(\mathbf{r}_1) \, \chi_q^*(\mathbf{r}_2) \, r_{12}^{-1} \chi_r(\mathbf{r}_1) \, \chi_s(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_q^*(\mathbf{r}_1) \, \chi_p^*(\mathbf{r}_2) \, r_{12}^{-1} \chi_s(\mathbf{r}_1) \, \chi_r(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_r^*(\mathbf{r}_1) \, \chi_s^*(\mathbf{r}_2) \, r_{12}^{-1} \chi_p(\mathbf{r}_1) \, \chi_q(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_s^*(\mathbf{r}_1) \, \chi_r^*(\mathbf{r}_2) \, r_{12}^{-1} \chi_q(\mathbf{r}_1) \, \chi_p(\mathbf{r}_2)
\end{aligned} \tag{5}$$

which in the bra–ket notation[11] becomes

$$\langle pq \mid rs \rangle = \langle qp \mid sr \rangle = \langle rs \mid pq \rangle = \langle sr \mid qp \rangle \tag{6}$$

and in the charge cloud notation[11] becomes

$$[pq \mid rs] = [rs \mid pq] = [qp \mid sr] = [sr \mid qp] \tag{7}$$

For real one-electron functions, there are further symmetries. They take the form

$$\begin{aligned}
&\int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_p(\mathbf{r}_1) \, \chi_q(\mathbf{r}_2) \, r_{12}^{-1} \chi_r(\mathbf{r}_1) \, \chi_s(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_r(\mathbf{r}_1) \, \chi_q(\mathbf{r}_2) \, r_{12}^{-1} \chi_p(\mathbf{r}_1) \, \chi_s(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_p(\mathbf{r}_1) \, \chi_s(\mathbf{r}_2) \, r_{12}^{-1} \chi_r(\mathbf{r}_1) \, \chi_q(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_r(\mathbf{r}_1) \, \chi_s(\mathbf{r}_2) \, r_{12}^{-1} \chi_p(\mathbf{r}_1) \, \chi_q(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_q(\mathbf{r}_1) \, \chi_p(\mathbf{r}_2) \, r_{12}^{-1} \chi_s(\mathbf{r}_1) \, \chi_r(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_s(\mathbf{r}_1) \, \chi_p(\mathbf{r}_2) \, r_{12}^{-1} \chi_q(\mathbf{r}_1) \, \chi_r(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_q(\mathbf{r}_1) \, \chi_r(\mathbf{r}_2) \, r_{12}^{-1} \chi_s(\mathbf{r}_1) \, \chi_p(\mathbf{r}_2) \\
&\quad = \int d\mathbf{r}_1 \int d\mathbf{r}_2 \, \chi_s(\mathbf{r}_1) \, \chi_r(\mathbf{r}_2) \, r_{12}^{-1} \chi_q(\mathbf{r}_1) \, \chi_p(\mathbf{r}_2)
\end{aligned} \tag{8}$$

which in bra–ket and charge cloud notations become

$$\begin{aligned}
\langle pq \mid rs \rangle &= \langle rq \mid ps \rangle = \langle ps \mid rq \rangle = \langle rs \mid pq \rangle \\
&= \langle qp \mid sr \rangle = \langle sp \mid qr \rangle = \langle qr \mid sp \rangle = \langle sr \mid qp \rangle
\end{aligned} \tag{9}$$

and

$$\begin{aligned}
[pq \mid rs] &= [pq \mid sr] = [qp \mid rs] = [qp \mid sr] \\
&= [rs \mid pq] = [rs \mid qp] = [sr \mid pq] = [sr \mid qp]
\end{aligned} \tag{10}$$

respectively.

Clearly, given a set of n one-electron functions, we can form n^4 two-

electron integrals. By taking account of the permutational symmetry properties for real orbitals this number is reduced to

$$n(n+1)(n^2+n+2)/8 \tag{11}$$

Figure 1 demonstrates the reduction in the size of the list of two-electron integrals that results from the exploitation of these permutational symmetries for a basis set of just two functions. In this figure a list of redundant integrals is presented along with the corresponding nonredundant list that results from the use of permutational symmetries. For basis sets of more realistic size the reduction in the length of the two-electron integral list obtained by recognizing these symmetries is even more dramatic, as can be seen in Table 1, where a comparison is made of the number of redundant and nonredundant integrals for various values of n, the number of functions in the basis set. The exploitation of the permutational symmetries of the indices reduces the external storage and input/output requirements but complicates the implementation of the four-index transformation algorithm.

2.3. Classification of Integrals

The two-electron integrals may be classified according to the number of centers they involve. If A, B, C,... denote functions centered at different points in space then clearly we can have one-center integrals

$$[AA|AA] \tag{12}$$

List of redundant integrals	*List of non-redundant integrals*
[11|11]	[11|11]
[21|11]	[21|11]
[12|11]	[21|21]
[22|11]	[22|11]
[11|21]	[22|21]
[21|21]	[22|22]
[12|21]	
[22|21]	
[11|12]	
[21|12]	
[12|12]	
[22|12]	
[11|22]	
[21|22]	
[12|22]	
[22|22]	

Figure 1. Reduction in the number of two-electron integrals resulting from the exploitation of permutational symmetry.

Table 1. Comparison of the Number of Redundant and Nonredundant Integrals for Various Values of n, the Number of Functions in the Basis Set

n	n^4	$n(n+1)(n^2+n+2)/8$
5	625	120
10	10000	1540
20	160000	22155
30	810000	108345
40	2560000	336610
50	6250000	813450
75	31640625	4062675
100	100000000	12753775
200	1600000000	202015050

two-center integrals

$$[AA|BB], \quad [AA|AB], \quad [AB|AB] \tag{13}$$

three-center integrals

$$[AA|BC], \quad [AB|BC] \tag{14}$$

and four-center integrals

$$[AB|CD] \tag{15}$$

The different types of two-center integrals given in (13) are sometimes termed Coulomb, hybrid, and exchange, respectively. The analysis of two-electron integrals in terms of the number of centers that they involve is crucial in the integral evaluation step of a molecular electronic structure calculation but is of little relevance to the four-index transformation stage, except in the case of the direct transformation approach[11] advocated in Section 7.3.

The two-electron integrals may also be divided into seven groups according to the various possible coincidences of the indices. This analysis is made in Figure 2. For each type of integral, the remaining integrals that can be obtained by permutation of the integrals are given. In some applications it is useful to sort the two-electron integrals according to the types defined in Figure 2, which may then be processed by different algorithms in the following calculation. For example, Billingsley[12] suggested segregating the two-electron integrals into seven types prior to the execution of a self-consistent-field program. This may also prove useful in the four-index transformation stage of a calculation.

Class	Integral	Equivalent integral types				
1	[ii|ii]					
2	[ij|ii]	[ji|ii]	[ii|ij]	[ii|ji]		
3	[ij|ij]	[ji|ij]	[ij|ji]	[ji|ji]		
4	[ii|jj]	[jj|ii]				
5	[ii|jk]	[ii|kj]	[jk|ii]	[kj|ii]		
6	[ij|ik]	[ij|ki]	[ji|ik]	[ji|ki]	[ik|ij]	
			[ik|ji]	[ki|ij]	[ki|ji]	
7	[ij|kl]	[ij|lk]	[ji|kl]	[ji|lk]	[kl|ij]	
			[kl|ji]	[lk|ij]	[lk|ji]	

Figure 2. Analysis of coincidences of indices for two-electron integrals.

A unique list of two-electron integral labels can be easily generated by employing the loop structure[11]

```
kount := 0
for i := 1 to n step 1
    for j := 1 to i step 1
        for k := 1 to i step 1
            if i = k then l_max := j else l_max := k
                for l := 1 to l_max step 1
                    kount := kount + 1
...
...
...
                next l
        next k
    next j
next i                                                    (16)
```

The index kount ranks the integrals in "canonical" order

$$i \geqslant j \tag{17a}$$

$$k \geqslant l \tag{17b}$$

$$ij \geqslant kl \tag{17c}$$

where

$$ij = [i(i-1)]/2 + j \qquad (i \geqslant j) \tag{18a}$$

and similarly

$$kl = [k(k-1)]/2 + 1 \qquad (k \geqslant l) \tag{18b}$$

ij \ kl	11	21	22	31	32	33	41	42	43	44	...
11	1	2	4	7	11	16	22	29	37	46	
21		3	5	8	12	17	23	30	38	47	
22			6	9	13	18	24	31	39	48	
31				10	14	19	25	32	40	49	
32					15	20	26	33	41	50	
33						21	27	34	42	51	
41							28	35	43	52	
42								36	44	53	
43									45	54	
44										55	
.											
.											
.											

Figure 3. The two-electron integral matrix. For each integral the value of the canonical index is given.

Thus the canonical index is

$$\text{kount} = [ij(ij-1)]/2 + kl \qquad (ij \geqslant kl) \tag{19}$$

which can be written in terms of i, j, k, and l as

$$\text{kount} = (i^4 - 2i^3 + 4i^2 j - i^2 - 4ij + 2i + 4j^2 - 4j + 2k^2 + 2k + 4l)/4 \tag{20}$$

The two-electron integrals can be arranged in the form of a Hermitian positive definite matrix with rows and columns labeled by products of two one-electron functions $\chi_i \chi_j$, $i \geqslant j$. The structure of the two-electron integral supermatrix is illustrated in Figure 3. In this figure, the value of the canonical index, kount, is indicated for each integral.

2.4. Packing of Integral Labels[11]

Each two-electron integral has associated with it four orbital indices that uniquely label it. If the integral list is dense, as is usually the case in atomic electronic structure calculations in which symmetry has been fully exploited, then all of the integrals can be stored in canonical order. The integral label is implied from its position in the list. Clearly, in this approach integrals whose value is zero have to be stored.

For molecular electronic structure studies, the integral list is, in general, sparce, particularly in the case of large molecules. In this case it is much more advantageous to store only the nonzero integrals together with the associated labels. However, in order to reduce the amount of disk storage required it is essential that the four indices associated with each of the integrals be packed into a single word of storage. Efficient procedures

```
C...  Pack using integer arithmetic
      INTEGER I, J, K, L, IJKL
      DATA NMAX /100/, N2 /10000/, N3 /1000000/
C...
      IJKL=I*N3 + J*N2 + K*NMAX + L
```

Figure 4. FORTRAN code for packing two-electron integral labels using integer arithmetic.

for packing and unpacking integral labels are therefore required in all phases of quantum chemical calculations including the four-index transformation. However, the efficiency of the packing procedure is particularly crucial in the four-index transformation phase since it involves both the list integrals over both the basis functions and the list of integrals over the molecular orbitals. Here we give an overview of integral label packing methods.

A portable, and to some extent machine independent, method of packing and unpacking integral labels is provided by integer arithmetic. Let NMAX be the maximum number of orbitals that may be considered by a given program. NMAX will be limited by the word length of the machine being used as we discuss further below. Let N2 and N3 be given by

$$\text{N2} = \text{NMAX} * \text{NMAX} \tag{21a}$$

$$\text{N3} = \text{NMAX} * \text{N2} \tag{21b}$$

The four labels I, J, K, L associated with a given integral can then be packed by means of the FORTRAN code shown in Figure 4. The integer variable $IJKL$ contains the packed integral label. Clearly, NMAX is restricted by (a) the maximum size of basis set one wishes to handle; (b) the requirement that $IJKL$ be stored in a single word. For example, on an IBM computer with four byte words, each byte consisting of eight bits, the largest integer that can be stored is $2^{(4\times 8)}-1$. Each label I, J, K, L can have a maximum value of 2^8-1, that is 255. An integral label that was packed by means of the code shown in Figure 4 can be unpacked by using the FORTRAN code displayed in Figure 5. Use is made of the fact that for integer division M/N is the largest integer less than M/N.

```
C...  Unpack using integer arithmetic
      INTEGER I, J, K, L, IJKL
      DATA NMAX /100/, N2 /10000/, N3 /1000000/
C...
      I = IJKL/N3
      J = (IJKL - I*N3)/N2
      K = (IJKL - (IJKL/N2)*N2)/NMAX
      L = IJKL - (IJKL/NMAX)*NMAX
```

Figure 5. FORTRAN code for unpacking two-electron integral labels using integer arithmetic.

```
C...  Pack using INTEGER*4 and LOGICAL*1 variables.
      INTEGER*4 I, J, K, L, IJKL
      LOGICAL*1 LI(4), LJ(4), LK(4), LL(4),LIJKL(4)
      EQUIVALENCE (I,LI(1)), (J,LJ(1)), (K,LK(1)),
     1            (L,LL(1)),(IJKL, LIJKL(1))
C...
      LIJKL(1) = LI(4)
      LIJKL(2) = LJ(4)
      LIJKL(3) = LK(4)
      LIJKL(4) = LL(4)
```

Figure 6. Integral label packing using INTEGER*4 and LOGICAL*1 variables on IBM machines.

On IBM machines, the four indices can be packed into an INTEGER*4 word by using LOGICAL*1 variables in conjunction with an EQUIVALENCE statement. FORTRAN code for packing an integral label using this approach is displayed in Figure 6. Again *I*, *J*, *K*, and *L* are the indices to be packed into *IJKL*. The packing of the integral label is illustrated in Figure 7. Each of the labels *I*, *J*, *K*, and *L* are packed into a single eight-bit byte and thus the maximum value that they can take is again 255. The corresponding FORTRAN code for unpacking a label is shown in Figure 8. In contrast to the packing method based on integer arithmetic shown in Figures 4 and 5, the method displayed in Figures 6, 7, and 8 requires no arithmetic operations and is therefore considerably faster.

On the CYBER 205 vector processing computer an extension of FORTRAN 77, known as FORTRAN 200, is employed. This FORTRAN dialect allows variables of type BIT. Each BIT variable consists of a single bit. Figure 9 shows code that can employed on the CYBER 205 using BIT arrays to pack four integral indices into a single 64-bit word. Figure 10 illustrates the packing process. By using a 64-bit word to pack four integral labels we are able to handle maximum values of $2^{16}-1$, which is far larger than is required. Greater efficiency can be achieved by packing eight labels to a 64-bit word. This allows labels up to a maximum of 255 to be handled. FORTRAN 200 code for unpacking labels that are packed using the algorithm given in Figure 9 is presented in Figure 11. For efficient implementation on a CYBER 205, or indeed any other vector processing computer, the label packing and unpacking process should be vectorized.

On many computers, including the CYBER 205 and the CRAY vector processing computers, labels can be efficiently packed using SHIFT together logical operations. For example, in Figure 12 FORTRAN code that can be implemented in the FPS 164 for packing integral labels is given. The packing process is illustrated in Figure 13. The corresponding FORTRAN code for unpacking the labels is given in Figure 14.

It should be clear from the above discussion that the efficient packing and unpacking of integral labels is to a large extent machine dependent.

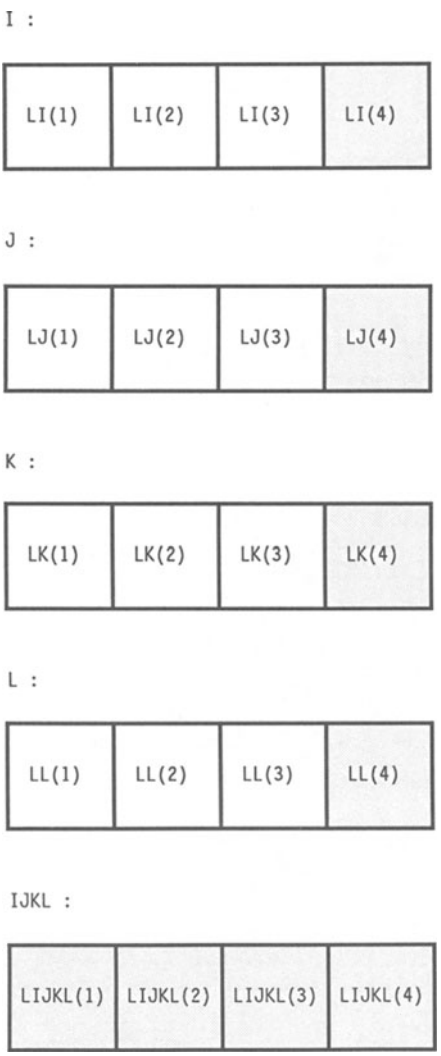

Figure 7. Illustration of the packing procedure given in Figure 6.

However, the range of examples given in this section should provide a basis for implementation on any computer. The first method described above is, to some extent, machine independent (although limited in the range of indices it can handle by the integer word length on a particular computer); it is also the least efficient technique. Increased efficiency usually entails the use of more machine-specific coding methods.[11]

```
C...  Unpack using INTEGER*4 and LOGICAL*1 variables
      INTEGER*4 I, J, K, L, IJKL
      LOGICAL*1 LI(4), LJ(4), LK(4), LL(4), LIJKL(4)
      EQUIVALENCE (I,LI(1)), (J,LJ(1)), (K,LK(1)),
     1            (L,LL(4)),(IJKL, LIJKL(1))
C...
      I=0
      J=0
      K=0
      L=0
      LI(4) = LIJKL(1)
      LJ(4) = LIJKL(2)
      LK(4) = LIJKL(3)
      LL(4) = LIJKL(4)
```

Figure 8. Integral label unpacking using INTEGER*4 and LOGICAL*1 variables on IBM machines.

2.5. Compressed Integral Lists

One of the major practical problems that arise in the execution of molecular electronic structure calculations is the need to store vast numbers of two-electron integrals, usually on a direct access peripheral storage. This problem arises in all stages of the calculations but can be particularly severe in the four-index transformation stage, involving as it does not only the list of integrals over the basis functions and the list of integrals over the molecular orbitals but also the list of half-transformed integrals. The severity of this problem can be reduced to some extent by employing compressed integral lists. This technique may reduce the size of a two-electron integral list by up to a factor of 3.[11]

In a typical application, the two-electron integral list can contain elements that differ by several orders of magnitude. However, the same length of word is usually employed to store these integrals irrespective of their size. The formation of a compressed integral list exploits the fact that some integrals are much smaller than others to reduce the amount of direct access peripheral storage required. Instead of employing a single buffer when reading or writing two-electron integrals, several buffers are main-

```
C...  Pack using INTEGER and BIT variables.
      INTEGER I, J, K, L, IJKL
      BIT BI(64), BJ(64), BK(64), BL(64), BIJKL(64)
      EQUIVALENCE (I,BI(1)), (J,BJ(1)), (K,BK(1)),
     1            (L,BL(1)),(IJKL,BIJKL(1))
C...
      BIJKL(1;16) = BI(49;16)
      BIJKL(17;16) = BJ(49;16)
      BIJKL(33;16) = BK(49;16)
      BIJKL(49;16) = BL(49;16)
```

Figure 9. Integral label packing using INTEGER and BIT variables on the CYBER 205 vector processor.

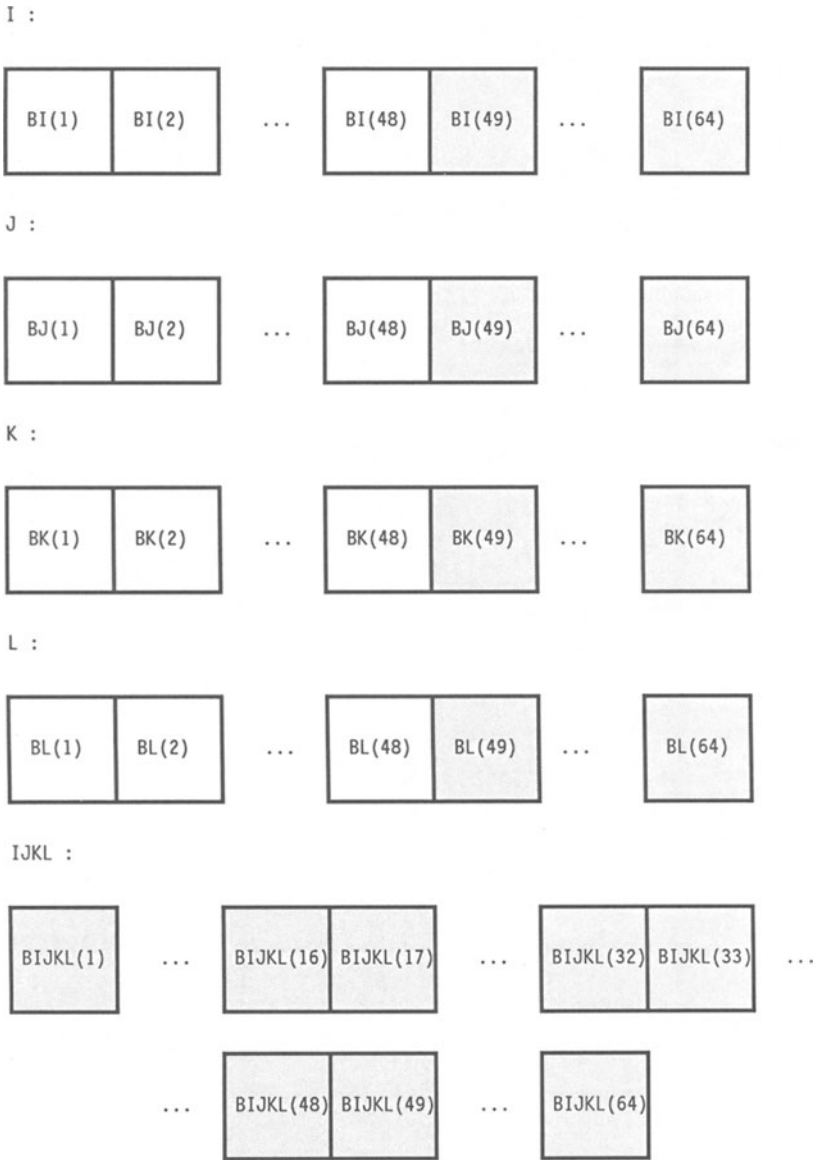

Figure 10. Illustration of the packing procedure given in Figure 9.

```
C...  Unpack using INTEGER and BIT variables.
      INTEGER I, J, K, L, IJKL
      BIT BI(64), BJ(64), BK(64), BL(64), BIJKL(64)
      EQUIVALENCE (I,BI(1)), (J,BJ(1)), (K,BK(1)),
     1            (L,BL(1)),(IJKL,BIJKL(1))
C...
      I=0
      J=0
      K=0
      L=0
      BI(49;16) = BIJKL(1;16)
      BJ(49;16) = BIJKL(17;16)
      BK(49;16) = BIJKL(33;16)
      BL(49;16) = BIJKL(49;16)
```

Figure 11. Integral label unpacking using INTEGER and BIT variables on the CYBER 205 vector processor.

tained and integrals are placed in them according to their size. The scheme is illustrated in Figure 15. When buffers containing small integrals are written to disk only the most significant bits in each word are retained. For large integrals the full word is retained. For each buffer the number of bits written to disk is related to the magnitude of the integrals it contains. The technique can be particularly useful when calculations are made for large molecules in which many two-electron integrals are small. It may also be useful when using universal even-tempered basis sets,[1] which are considered in more detail in Section 6.3.

3. Four-Index Transformations

3.1. Orbital Transformations

Let two sets of one-electron functions, ϕ, $i = 1, 2, ..., m$, and χ, $p = 1, 2, ..., n$ be related by the linear transformation

$$\phi_i = \sum_{p=1}^{n} \chi_p T_{ip}, \qquad i = 1, 2, ..., m \tag{22}$$

which may be written as a vector-matrix product as

$$\boldsymbol{\phi} = \boldsymbol{\chi}\mathbf{T} \tag{23}$$

```
C...  Pack using SHIFT and OR.
      INTEGER I, J, K, L, IJKL
C...
      IJKL = OR(SHIFT(I,8),J)
      IJKL = OR(SHIFT(IJKL,8),K)
      IJKL = OR(SHIFT(IJKL,8),L)
```

Figure 12. Integral label packing using SHIFT and OR on the FPS 164.

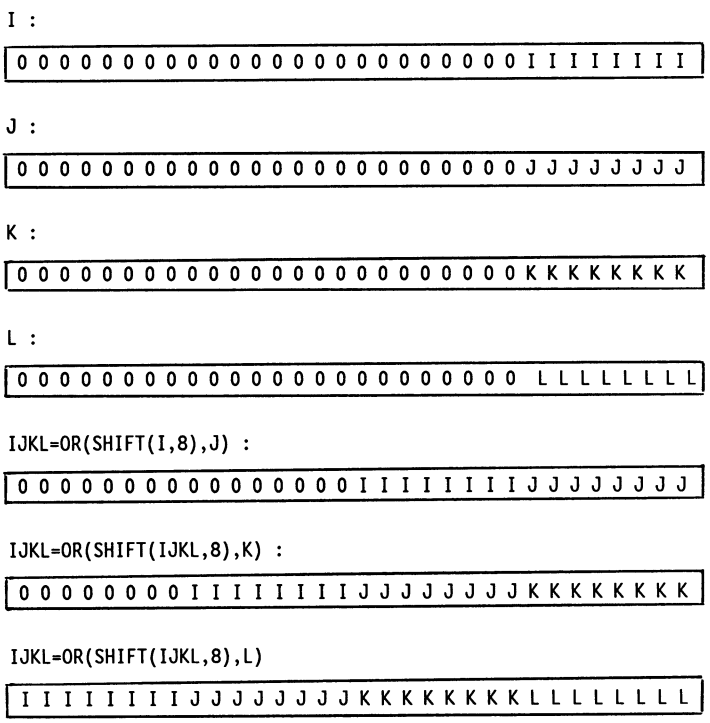

Figure 13. Illustration of the packing procedure given in Figure 12. Each symbol represents a bit.

where T is an $m \times n$ matrix of coefficients and ϕ and χ are the molecular orbitals and the basis functions, respectively. The coefficients, T_{ip}, will usually result from a self-consistent-field calculation but may, for example, be derived from an orthogonalization procedure or symmetry adaption. In this section, we address the central aspect of this chapter, the transformation of integrals over the functions χ_p to integrals over the functions ϕ_i. One-electron integral transformations are considered in Section 3.2 and

```
C...  Unpack using SHIFT and AND
      INTEGER I, J, K, L, IJKL
C...
      L = AND(IJKL,O'377')
      IJKL = SHIFT(IJKL,-8)
      K = AND(IJKL,O'377')
      IJKL = SHIFT(IJKL,-8)
      J = AND(IJKL,O'377')
      IJKL = SHIFT(IJKL,-8)
      I = AND(IJKL,O'377')
```

Figure 14. Integral label unpacking using SHIFT and AND on the FPS 164.

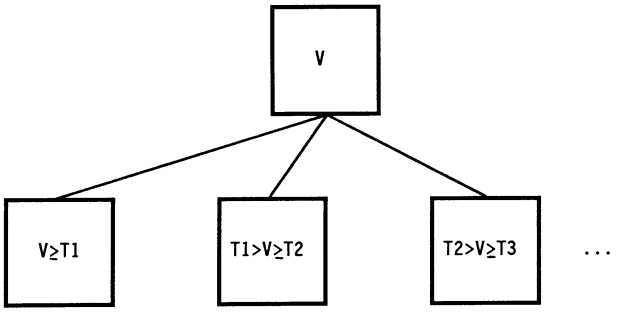

Figure 15. The use of compressed integral lists. V denotes the absolute value of a two-electron integral and T1, T2, T3,... define the tolerances used to divide the integral list.

two-electron integral transformations are considered in Sections 3.3, 3.4, 3.5, and 3.6. The integral sorting method due to Yoshimine[13] is considered in Section 3.7.

3.2. One-Electron Integral Transformation

The one-electron integral transformation may be written

$$\langle \phi_i | h | \phi_j \rangle = \sum_{p=1}^{n} \sum_{q=1}^{n} T_{ip}^{\dagger} \langle \chi_p | h | \chi_q \rangle T_{jq} \tag{24}$$

or as the matrix product

$$\mathsf{H} = \mathsf{T}^{+} \mathsf{h} \mathsf{T} \tag{25}$$

where T is an $m \times n$ matrix of coefficients, H is the matrix $\langle \boldsymbol{\phi} | h | \boldsymbol{\phi} \rangle$, and h is the matrix $\langle \boldsymbol{\chi} | h | \boldsymbol{\chi} \rangle$ for some one-electron operator h. This may be termed a two-index transformation. Computationally, this transformation is insignificant in comparison with the two-electron integral transformation. The entire one-electron integral transformation can be achieved in core and requires n^3 operations.

3.3. Two-Electron Integral Transformation

Now consider the transformation of the two-electron integrals over the functions χ, which we denote by $[pq|rs]$, to the corresponding integrals over the functions ϕ, which we denote by $[ij|kl]$. This involves the four-index transformation which is the concern of this chapter.

The four-index transformation can be written in its most elementary form as

$$[ij|kl] = \sum_{p=1}^{n} \sum_{q=1}^{n} \sum_{r=1}^{n} \sum_{s=1}^{n} T_{ip}^{\dagger} T_{kr}^{\dagger} [pq|rs] T_{jq} T_{ls} \tag{26}$$

where we have assumed for simplicity that $m = n$. Clearly, since there are of the order of n^4 two-electron integrals, the number of terms in these summations increases as the eighth power of the number of basis functions. Direct application of (26) would, therefore, rapidly become computationally prohibitive as the size of the basis set increases.

It is evident that a much more efficient scheme can be devised by performing four partial transformations

$$[iq|rs] = \sum_{p=1}^{n} T_{ip}^{\dagger}[pq|rs], \qquad i, q, r, s = 1,..., n \tag{27a}$$

$$[ij|rs] = \sum_{q=1}^{n} [iq|rs]\, T_{jq}, \qquad i, j, r, s = 1,..., n \tag{27b}$$

$$[ij|ks] = \sum_{r=1}^{n} T_{kr}^{\dagger}[ij|rs], \qquad i, j, k, s = 1,..., n \tag{27c}$$

and

$$[ij|kl] = \sum_{s=1}^{n} [ij|ks]\, T_{ls}, \qquad i, j, k, l = 1,..., n \tag{27d}$$

Each of these partial transformations involves $\sim n^5$ terms and the full transformation, therefore, requires $\sim 4n^5$ terms. The first published description of this scheme is due to Tang and Edmiston.[14]

Alternatively, the four-index transformation may be regarded as two two-index transformations

$$[pq|kl] = \sum_{r=1}^{n} \sum_{s=1}^{n} T_{kr}^{\dagger}[pq|rs]\, T_{ls}, \qquad p, q, k, l = 1,..., n \tag{28a}$$

and

$$[ij|kl] = \sum_{p=1}^{n} \sum_{q=1}^{n} T_{ip}^{\dagger}[pq|kl]\, T_{jq}, \qquad i, j, k, l = 1,..., n \tag{28b}$$

Each of these two-index transformations involves $n^2(2n^3)$ terms and so the full transformation again involves $\sim 4n^5$ terms.

We note that the partially transformed integrals $[pq|kl]$ are not, in general, in the same order as they are required for the second two-index transformation. The necessary reordering, which we discuss further below, can be achieved by an algorithm that depends on the fourth power of the number of basis functions.

A transformation algorithm, published in 1963 by Nesbet,[15] should briefly be mentioned. If we form the products of coefficients

$$C_{pq}^{ij} = T_{ip}^{\dagger} T_{jq} \tag{29a}$$

and

$$C_{rs}^{kl} = T_{kr}^{\dagger} T_{ls} \tag{29b}$$

then the four-index transformation may be written

$$[ij|kl] = \sum_{p=1,q=1} \sum_{r=1,s=1} C_{pq}^{ij} [pq|rs] C_{rs}^{kl} \tag{30}$$

It should be clear that this approach depends on the sixth power of the number of basis functions.

In this section, we consider three transformation algorithms in some detail. Section 3.4 describes the Bender–Shavitt algorithm. This algorithm is typical of a number of schemes that were devised in the period 1970–1977. The Elbert algorithm, which was first published in 1978, is considered in Section 3.5, and the algorithm of Saunders and van Lenthe is described in Section 3.6. The integral sorting, which is necessary in all of the schemes described below, is addressed in Section 3.7.

3.4. The Bender–Shavitt Method

The first four-index transformation algorithm that we explicitly consider in this article is that due to Bender[3] and Shavitt.[4] We present it primarily as an example of early approaches to the problem of devizing an efficient four-index transformation. The reader should also note the work of McLean,[16] of Yoshimine,[17] of Bagus *et al.*,[18] of Diercksen,[19] of Pounder,[20] and of Pendergast and Fink.[21]

The loop structure suggested by Bender and Shavitt is shown in Figure 16. In this figure $g(p, q, r, s)$ is the list of integrals over the basis functions and is assumed to be in canonical order. The only core storage required for the $g(p, q, r, s)$ integrals is an input buffer. The list $g(p, q, r, s)$ is read into core only once. The integrals are assumed to be over real orbitals and the symmetry number, e, which can take values of 1, 2, 4, or 8 depending upon equalities among the indices, takes account of all eight allowed permutations of the four orbital labels which identify the integral. The arrays s1, s2, s3, s4a, and s4b are used to store partially transformed integral lists. s1 and s3 can easily be held in core. The half-transformed integral list, s2, is stored on a direct access peripheral device as are the lists of fully transformed integrals, s4a and s4b. In Figure 16, $G(i, j, k, l)$ denotes the final list of fully transformed integrals.

Let us now analyze, in some detail, the number of multiplications required in each stage of the Bender–Shavitt algorithm for the four-index transformation. If n is the number of basis functions and m is the number

```
for p:=1 to n step 1
     e:=1
     for q:=1 to p step 1
          s2(p,q,-,-):=0.0
          if p=q then e:=2
          for r:=1 to p step 1
               s1(-):=0.0
               if p=r then smax=q else smax=r
               for s:=1 to smax step 1
                    if r=s then e:=e+e
                    if (pq)=(rs) then e:=e+e
                    gs:=g(p,q,r,s)/e
                    for i:=1 to m step 1
                         s1(i):=s1(i) + c(s,i)*gs
                    next i
               next s
               for i:=1 to m step 1
                    si:=s1(i)
                    cri:=c(r,i)
                    for j:=1 to i step 1
                         s2(p,q,i,j):=s2(p,q,i,j) + cri*s1(j)
                                                  + c(r,j)*si
                    next j
               next i
          next r
     next q
next p


for i:=1 to m step 1
     for j:=1 to i step 1
          for p:=1 to n step 1
               s3(-):=0.0
               for q:=1 to p step 1
                    spq:=s2(p,q,i,j)
                    for k:=1 to m step 1
                         s3(k):=s3(k) + c(q,k)*spq
                    next k
               next q
               s4a(i,j,-,-):=0.0
               for k:=1 to i step 1
                    cpk:=c(p,k)
                    sk:=s3(k)
                    if i=k then
                         begin
                              lmin:=i
                              lmax:=j
                         end
                    else
                         begin
                              lmin:=1
                              lmax:=k
                         end
                    for l:=lmin to lmax step 1
                         s4a(i,j,k,l):=s4a(i,j,k,l) + cpk*s3(l) + c(p,l)*sk
                    next l
               next k
               s4b(-,-,i,j):=0.0
               for ip:=1 to m step 1
                    cpi:=c(p,ip)
                    si:=s3(p,ip)
                    if i=ip then
                         begin
```

Figure continued

```
                              jpmin:=j
                              jpmax:=i
                         end
                    else
                         begin
                              jpmin:=1
                              jpmax:=ip
                         end
                    for jp:=jpmin to jpmax step 1
                         s4b(i,j,i,j):=s4b(i,j,i,j) + cpi*s3(jp)
                                                    + c(p jp)*si
                    next jp
               next ip
          next p
     next j
next i


for i:=1 to m step 1
     for j:=1 to i step 1
          for k:=1 to i step 1
               if i=k then
                    begin
                         lmin:=i
                         lmax:=j
                    end
               else
                    begin
                         lmin:=1
                         lmax:=k
                    end
               for l:=lmin to lmax step 1
                    G(i,j,k,l):=s4a(i,j,k,l) + s4b(i,j,k,l)
               next l
          next k
     next j
next i
```

Figure 16. The loop structure suggested by Bender and Shavitt.

of molecular orbitals, then incorporation of the symmetry number, e, requires

$$[n(n+1)(n^2+n+2)]/8 \tag{31}$$

multiplications. Formation of the one-quarter transformed integral list, s1, requires

$$[mn(n+1)(n^2+n+2)]/8 \tag{32}$$

multiplications, while formation of the half transformed integrals, s2, requires a further

$$[m(m+1)\,n(2n^2+3n+1)]/6 \tag{33}$$

multiplications. s3, the three-quarters transformed integral list, can be formed by a further

$$[m^2(m+1)\,n(n+1)]/4 \tag{34}$$

multiplications. Finally, formation of the fully transformed integral lists, s4a and s4b, can be achieved in

$$[mn(m+1)(m^2+m+2)]/4 \tag{35}$$

and

$$[m^2(m+1)(m^2+m+2)]/4 \tag{36}$$

multiplications, respectively.

In common with the algorithms of Diercksen,[19] of Pounder,[20] of Pendergast and Fink,[21] and others, the Bender–Shavitt algorithm scales as

$$29n^5/24 \tag{37}$$

in the case $m=n$. It was not until 1978 that a significant improvement to these early algorithms was suggested, and it to this that we now turn our attention.

3.5. The Elbert Method

In 1978, Elbert[5] proposed an algorithm for the four-index transformation for which, in the case $m=n$, the number of multiplications increases as

$$(25n^5+42n^4+23n^3+6n^2)/24 \tag{38}$$

The loop structure for Elbert's algorithm is displayed in Figure 17. Again, s1, s2, and s3 denote arrays of partially transformed integrals. The half-transformed integrals, s2, are stored on a direct access peripheral device. To explain the superiority of Elbert's algorithm, we examine the number of multiplications involved in each step. The transformation over the first index involves

$$n^2[n(n+1)/2]\,m \tag{39}$$

```
for r:=1 to n step 1
    for s:=1 to r step 1
        for p:=1 to n step 1
            gpp:=g(p,p,r,s)
            for i:=1 to m step 1
                s1(p,i):=c(p,i)*gpp
            next i
            for q:=1 to p-1 step 1
                gpq:=g(p,q,r,s)
                for i:=1 to m step 1
                    s1(q,i):=s1(q,i) + c(p,i)*gpq
                    s1(p,i):=s1(p,i) + c(q,i)*gpq
                next i
            next q
        next p
        s2(r,s,-,-):=0.0
        for i:=1 to m step 1
            for q:=1 to n step 1
                sqi:=s1(q,i)
                for j:=1 to i step 1
                    s2(r,s,i,j):=s2(r,s,i,j) + c(q,j)*sqi
                next j
            next q
        next i
    next s
next r


for i:=1 to m step 1
    for j:=1 to i step 1
        for r:=1 to n step 1
            srr:=s2(r,r,i,j)
            for k:=i to m step 1
                s3(r,k):=c(r,k)*srr
            next k
            for s:=1 to r-1 step 1
                srs:=s2(r,s,i,j)
                for k:=i to m step 1
                    s3(s,k):=s3(s,k) + c(r,k)*srs
                    s3(r,k):=s3(r,k) + c(s,k)*srs
                next k
            next s
        next r
        G(i,j,-,-):=0.0
        for k:=i to m step 1
            for s:=1 to n step 1
                ssk:=s3(s,k)
                if i=k then lmin:=j else lmin:=1
                for l:=lmin to k step 1
                    G(i,j,k,l):=G(i,j,k,l) + c(s,l)*ssk
                next l
            next s
        next k
    next j
next i
```

Figure 17. The loop structure suggested by Elbert.

multiplications, which reduces to

$$n^4(n+1)/2 \tag{40}$$

when $m = n$. The transformation over the second index requires

$$n[n(n+1)/2]\, m(m+1)/2 \tag{41}$$

multiplications, which becomes

$$n^3(n+1)^2/4 \tag{42}$$

when $m = n$. It is the transformation over the third index that departs from earlier formulations of the four-index transformation in Elbert's algorithm. This involves

$$n^2[m(m+1)/2](m+1) - m(2m^2+3m+1)/6 \tag{43}$$

multiplications or

$$n^3(n+1)(n+2)/6 \tag{44}$$

in the case $m = n$. The final transformation requires

$$nm(m+1)(m^2+m+2)/8 \tag{45}$$

multiplications, reducing to

$$n^2(n^3+3n^2+3n+2)/8 \tag{46}$$

in the case $m = n$. The total number of multiplications is therefore

$$n^4m + 6n^3m(m+3) + 2n^2m(m+1)(2m+7) + 3nm(m^3+2m^2+3m+2)/24 \tag{47}$$

which reduces to the expression (38) given above when $m = n$.

The n dependence of each step of the transformation is illustrated in Table 2. The number of multiplications involved in each step of the Bender–Shavitt algorithm is given for comparative purposes. For a basis set of 30 functions, for example, the Elbert algorithm involves only 85% of the multiplications required in the Bender–Shavitt approach. We underline the increased efficiency of the Elbert algorithm by comparing the orthodox loop structure (16) for generating a complete permutationally non-redundant list of two-electron integrals, which we rewrite as

```
for p := 1 to n step 1
    [Executed n times]
        for q := 1 to p step 1
            [Executed n^2/2 times]
                for r := 1 to p step 1
                    [Executed n^3/3 times]
                        if p = r then smax = q else smax = r
                            for s := 1 to smax step 1
                                [Executed n^4/8 times]
                                ...
                                ...
                                ...
                            next s
                next r
        next q
next p                                                    (48)
```

with the Elbert loop structure

```
for p := 1 to n step 1
    [Executed n times]
        for q := 1 to p step 1
            [Executed n^2/2 times]
                for r := p to n step 1
                    [Executed n^3/6 times]
                        if p = r then smax = q else smax = r
                            for s := 1 to smax step 1
                                [Executed n^4/4 times]
                                ...
                                ...
                                ...
                            next s
                next r
        next q
next p                                                    (49)
```

The third loop (with index r) has an average length of $2n/3$ and $n/3$ in the orthodox and in the Elbert schemes, respectively.

3.6. The Saunders–van Lenthe Method

An algorithm that is more efficient than that of Elbert[5] when the number of transformed functions, m, is less than the number of original functions, n, was proposed in 1983 by Saunders and van Lenthe.[6] The

Table 2. Number of Multiplications in Each Step of the Four-Index Transformation Algorithm Suggested by Elbert for the Case $m = n$[a]

n	Step 1	Step 2	Step 3	Step 4	Total
5	1875	1125	875	600	4475
	(720)	(1650)	(1125)	(2400)	(5895)
10	55000	30250	22000	15400	122650
	(16940)	(42350)	(30250)	(61600)	(151140)
20	1680000	882000	616000	443100	3621100
	(465255)	(1205400)	(882000)	(1772400)	(4325055)
30	12555000	6486750	4464000	3250350	26756100
	(3358695)	(8793150)	(6486750)	(13001400)	(31639995)
40	52480000	26896000	18368000	13464400	111208400
	(13801010)	(36309600)	(26896000)	(53857600)	(130864210)
50	159375000	81281250	55250000	406725000	336578750
	(41485950)	(109458750)	(81281250)	(162690000)	(394915950)

[a] The numbers of multiplications in each step of the Bender–Shavitt algorithm are given in parentheses.

loop structure for the Saunders–van Lenthe algorithm is shown in Figure 18. Before discussing this algorithm further, we shall describe how a more efficient procedure can be attained in the case $m < n$.

Consider the two-index transformation

$$\mathsf{B} = \mathsf{X}^\dagger \mathsf{A} \mathsf{X} \tag{50}$$

where A is an $n \times n$ symmetric matrix, B is an $m \times m$ symmetric matrix, and X is an $m \times n$ matrix of expansion coefficients. B may be evaluated as follows:

$$\mathsf{P} = \mathsf{A} \mathsf{X} \tag{51}$$

$$\mathsf{B} = \mathsf{X}^\dagger \mathsf{P} \tag{52}$$

The formation of B requires

$$mn^2 + m^2 n/2 \tag{53}$$

multiplication operations. Now if we write A in the form

$$\mathsf{A} = \mathsf{L} + \mathsf{L}^\dagger \tag{54}$$

where L is the lower triangular matrix

$$\begin{aligned} L_{ij} &= A_{ij}/2^{\delta_{ij}} \qquad \text{if} \quad i \geqslant j \\ &= 0 \qquad\qquad\quad \text{if} \quad i < j \end{aligned} \tag{55}$$

```
for p:=1 to n step 1
    for q:=1 to p step 1
        for r:=p to n step 1
            if p=r then smin:=q else smin:=1
            for s:=smin to r step 1
                for i:=1 to m step 1

                H(pqri) = ∑_s G(pqrs) Q(si)

                next i
            next s
        next r
        for r:=p to n step 1
            for i:=1 to m step 1
                for j:=1 to m step 1

                I(pqji) = ∑_r H(pqri) Q(rj)

                next j
            next i
        next r
        for i:=1 to m  step 1
            for j:=1 to i step 1

            J(pqji) = I(pqji) + I(pqij)

            next j
        next i
    next q
next p
for i:=1 to m step 1
    for j:=1 to i step 1
        for p:=1 to n step 1
            for q:=1 to p step 1
                for k:=1 to m step 1

                K(pkji) = ∑_q J(pqji) Q(qk)

                next k
            next q
        next p
        for p:=1 to n step 1
            for k:=1 to m step 1
                for l:=1 to m step 1

                L(lkji) = ∑_p K(pkji) Q(pl)

                next l
            next k
        next p
        for k:=1 to m step 1
            for l:=1 to k step 1

            M(lkji) = L(lkji) + L(klji)

            next l
        next k
    next j
next i
```

Figure continued

```
mm:=(m*(m+1))/2
for ji:=1 to mm step 1
     for lk:=1 to ji step 1

     [lkji] = M(lkji) + M(jilk)

     next lk
next ji


Notes.

(i) The integrals over basis functions are scaled according to

          G(pqrs) = [pqrs]/e

where e=1, 2, 4 or 8 depending upon coincidences amongst the indices.  The
integrals are ordered so that for a given pq (p>q) all rs (r>s and rs>pq)
are available.

(ii) The J(pqji)'s are ordered so that for a given ji (j≤i) all pq (p>q) are
available.

(iii) The M(jilk)'s are ordered so that for a given ji all lk (lk≤ji) are
available.
```

Figure 18. The loop structure suggested by Saunders and van Lenthe for the complete four-index transformation.

B may be evaluated as follows:

$$\mathsf{P} = \mathsf{LX} \tag{56}$$

$$\mathsf{Q} = \mathsf{X}^{\dagger}\mathsf{P} \tag{57}$$

and

$$\mathsf{B} = \mathsf{Q} + \mathsf{Q}^{\dagger} \tag{58}$$

The formation of B requires

$$mn^2/2 + m^2n \tag{59}$$

multiplications. It is this second approach to the two-index transformation (28) that is exploited in the algorithm of Saunders and van Lenthe when $m < n$.

The loop structure advocated by Saunders and van Lenthe is illustrated in Figure 18. As for the previous algorithms, s1, s2, and s3 denoted arrays of partially transformed integrals. The half-transformed integrals, s2, are stored on a direct access peripheral device. Let us consider the number of floating point multiplications required in each of the steps of

Table 3. Comparison of the Number of Multiplications in the Algorithms of Elbert and of Saunders and van Lenthe

n	m	Elbert algorithm	Saunders–van Lenthe algorithm
20	10	1091666	533333
	20	3333333	3333333
50	25	106608072	52083333
	30	138187500	79312500
	40	217666666	168583333
	50	325520833	325520833

n is the total number of basis functions, m is the number of active orbitals.

the algorithm displayed in Figure 18. Transformation over the first index involves

$$mn^4/8 \tag{60}$$

multiplications, while the second and third index transformations require

$$m^2n^3/6 \tag{61}$$

and

$$m^3n^2/4 \tag{62}$$

multiplications, respectively. Transformation over the fourth index requires

$$m^4n/2 \tag{63}$$

multiplications, so that the total number of multiplications required is

$$mn^4/8 + m^2n^3/6 + m^3n^2/4 + m^4n/2 \tag{64}$$

Both the Elbert and the Saunders–van Lenthe algorithms require $\sim 25n^5/24$ multiplications in the case $m = n$. The number of multiplications in the two algorithms for typical values of m and n are displayed in Table 3. This Table clearly demonstrates the superiority of the Saunders–van Lenthe algorithm when $m < n$.

3.7. Integral Sorting

The algorithms for the four-index transformation described in the preceding sections can be implemented straightforwardly if the input array for summing over a given index is totally available in fast memory. However, this is not usually the case for any but the very smallest basis sets, and therefore on completion of one step the results have to be reor-

dered for efficient processing in the following step. Yoshimine[13] devised an efficient sorting algorithm using direct access periheral storage that can effect this reordering, and which we now describe.

Assume that we have a list of two-electron integrals with labels $[pq|rs]$, which we take to be randomly ordered. The number of values of the compound index (rs) is $n(n+1)/2$, where n is the number of basis functions, for each value of the index (pq). Determine the range of values of (pq) for which all values of (rs) can be held in core. We call this a core load. Let N be the number of core loads that are required to handle the complete integral list.

The first step of the reordering process is to divide the available core into N bins, one associated with each core load. Now we consider the integrals $[pq|rs]$, one at a time, and decide which bin they belong to. This process is illustrated in the upper part of Figure 19. Each time a bin is filled

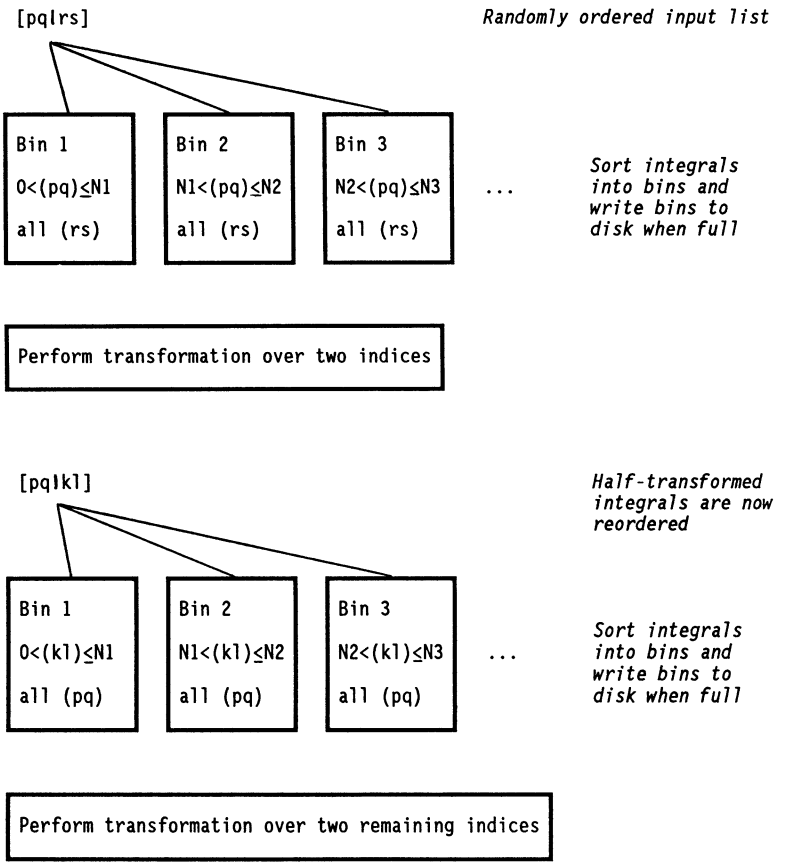

Figure 19. The Yoshimine sort.

it is written onto disk and an index maintained of the disk addresses of the filled bins. When this process is completed, we can return all of the bins associated with the first core load to core and carry out the summations over r and s for the particular range of p and q associated with that core load. This process is repeated for all N core loads.

The next stage of the computation is to reorder the integrals $[pq|kl]$. This is performed in the same manner as for the first two indices. Some partial order may exist in the half-transformed list of integrals, which can be exploited to minimize the testing and logical operations. The second reordering in the four-index transformation is illustrated in the lower part of Figure 19. To be effective the Yoshimine sorting algorithm reqires the use of a reasonable size buffer for each bin. If the buffer length is equated with the track length on the direct access disk, rotational delay can be avoided. For a long integral list the buffers may require a large amount of memory.

4. Point Group Symmetry

4.1. Point Group Symmetry in Quantum Chemistry

Point group symmetry can be exploited in almost all stages of atomic and molecular electronic structure calculations. If a basis set of symmetry adapted functions is employed then the number of integrals that have to be processed and stored can be reduced. This reduction is particularly significant in the case of the two-electron integrals. The two-electron integral

$$\langle \chi_p(\Gamma^{(p)})\, \chi_q(\Gamma^{(q)})\, |r_{12}^{-1}|\, \chi_r(\Gamma^{(r)})\, \chi_s(\Gamma^{(s)}) \rangle \tag{65}$$

where $\chi_p(\Gamma^{(p)})$ denotes a basis function that transforms according to the $\Gamma^{(p)}$ irreducible representation of the point symmetry group to which the system being studied belongs, will only be nonzero if the direct product

$$\Gamma^{(p)} \otimes \Gamma^{(q)} \otimes \Gamma^{(r)} \otimes \Gamma^{(s)} \tag{66}$$

contains the totally symmetric A_1 irreducible representation.

Molecular symmetry can significantly reduce the amount of computation required to effect a four-index transformation. Clearly, the greatest efficiency will be achieved when both the basis functions and the molecular orbitals are symmetry adapted with respect to the molecular point symmetry group. In general, we have

$$\langle \phi_i(\Gamma^{(i)})\, \phi_j(\Gamma^{(j)})\, |r_{12}^{-1}|\, \phi_k(\Gamma^{(k)})\, \phi_l(\Gamma^{(l)}) \rangle$$
$$= \sum_p \sum_q \sum_r \sum_s T^{\dagger}_{ip}(\Gamma^{(i)})\, T^{\dagger}_{jq}(\Gamma^{(j)})\, T_{kr}(\Gamma^{(k)})\, T_{ls}(\Gamma^{(l)})$$
$$\times \langle \chi_p(\Gamma^{(i)})\, \chi_q(\Gamma^{(j)})\, |r_{12}^{-1}|\, \chi_r(\Gamma^{(k)})\, \chi_s(\Gamma^{(l)}) \rangle \tag{67}$$

where χ denotes a basis function and ϕ denotes a molecular orbital.

When exploiting spatial symmetry in atomic and molecular electronic structure calculations, it should be noted that when $\Gamma^{(i)} = \Gamma^{(j)} = \Gamma^{(k)} = \Gamma^{(l)}$, we can impose the restrictions $i \geqslant j$, $k \geqslant l$ and $(ij) \geqslant (kl)$ the remaining integrals being obtained by means of the permutational symmetries considered in Section 2.2. However, when, for example, $\Gamma^{(i)} \neq \Gamma^{(j)}$, we must then consider all values of i and j and p and q. The different cases that can occur may be summarized as follows:

$$ij = ji, \quad kl = lk, \quad (ij) = (kl) \tag{68a}$$
$$ij \neq ji, \quad kl = lk, \quad (ij) = (kl) \tag{68b}$$
$$ij = ji, \quad kl = lk, \quad (ij) \neq (kl) \tag{68c}$$
$$ij \neq ji, \quad kl \neq lk, \quad (ij) = (kl) \tag{68d}$$
$$ij \neq ji, \quad kl = lk, \quad (ij) \neq (kl) \tag{68e}$$
$$ij = ji, \quad kl \neq lk, \quad (ij) \neq (kl) \tag{68f}$$
$$ij \neq ji, \quad kl \neq lk, \quad (ij) \neq (kl) \tag{68g}$$

In this section we discuss the exploitation of spatial symmetry in the four-index transformation stage of a quantum chemical calculation. In Section 4.2, we consider atoms, in Section 4.3 diatomic and linear molecules, and in Section 4.4 polyatomic molecules.

4.2. Atomic Systems

The spherical symmetry of atomic systems can be used to simplify electronic structure calculations considerably. Frequently, in atomic calculations all of the two-electron integrals that are not zero for symmetry reasons are stored. Integral labels can then be dispensed with since they are implied from the position of the integral in the list.

For atoms it is most profitable to use basis functions that consist of a radial function multiplied by a complex spherical harmonic

$$R_{nl}(r)\, Y_{lm}(\theta, \phi) \tag{69}$$

The quantum numbers l and m are associated with the symmetry group of

First charge distribution	Second charge distribution
ss	ss
ps	ps
pp	pp, ss
ds	pp, ds
dp	dp
dd	dd, pp, ss
fs	fs, dp
fp	fp, dd, ds, pp
fd	fd, dp, ps
ff	ff, dd, pp, ss
gs	gs, fp, dd
gp	gp, fd, fs, dp
gd	gd, ff, fp, ds, pp
gf	gf, fd, dp, ps
gg	gg, ff, dd, pp, ss

Figure 20. Symmetry blocking of two-electron integrals for atomic systems.

the free atom, the rotation-reflection group O_3. The symmetry blocking of the two-electron integral supermatrix that results from this symmetry is summarized in Figure 20.

4.3. Diatomic and Linear Molecules

Diatomic and linear molecules belong to the axial point symmetry groups $C_{\infty v}$ and $D_{\infty h}$. The character table for the group $C_{\infty v}$ is given in Table 4 and the character table for $D_{\infty h}$ group is given in Table 5.

Basis functions for diatomic and linear molecules usually include a factor depending on the axial angle, ϕ, which is of the form $\exp(im\phi)$. The basis functions are labeled σ, π, δ, ... according to the value of m. The two-electron integral $[ij|kl]$ can be shown to be zero for reasons of symmetry unless

$$(m_i - m_j) = -(m_k - m_l) \tag{70}$$

Hence the nonzero blocks of two-electron integrals that can arise are as summarized in Figure 21.

Table 4. Character Table for the $C_{\infty v}$ Point Symmetry Group

$C_{\infty v}$	E	$2C_{\infty}^{\phi}$	$\cdots$	$\infty\sigma_v$
Σ^+	1	1	$\cdots$	1
Σ^-	1	1	$\cdots$	-1
Π	2	$2\cos\phi$	$\cdots$	0
Δ	2	$2\cos 2\phi$	$\cdots$	0
Φ	2	$2\cos 3\phi$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

For molecules having $D_{\infty h}$ symmetry additional symmetry with respect to inversion is present. The basis functions are given an additional label, gerade (g) or ungerade (u), according to their symmetry properties under inversion. The four possibilities that arise are

$$\begin{gathered}[gg\,|\,gg] \\ [gg\,|\,uu] \\ [gu\,|\,gu] \\ [uu\,|\,uu]\end{gathered} \tag{71}$$

4.4. Polyatomic Molecules

Large polyatomic molecules are seldom symmetrical. For this reason some molecular electronic structure programs make no use of point group symmetry. Other programs exploit limited symmetry elements such as mirror planes. Programs that use the point group symmetry fully can make impressive calculations on large symmetrical molecules.

Table 5. Character Table for the $D_{\infty h}$ Point Symmetry Group

$D_{\infty h}$	E	$2C_{\infty}^{\phi}$	$\cdots$	$\infty\sigma_v$	i	$2S_{\infty}^{\phi}$	$\cdots$	∞C_2
Σ_g^+	1	1	$\cdots$	1	1	1	$\cdots$	1
Σ_g^-	1	1	$\cdots$	-1	1	1	$\cdots$	-1
Π_g	2	$2\cos\phi$	$\cdots$	0	2	$-2\cos\phi$	$\cdots$	0
Δ_g	2	$2\cos 2\phi$	$\cdots$	0	2	$2\cos 2\phi$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Σ_u^+	1	1	$\cdots$	1	-1	-1	$\cdots$	-1
Σ_u^-	1	1	$\cdots$	-1	-1	-1	$\cdots$	1
Π_u	2	$2\cos\phi$	$\cdots$	0	-2	$2\cos\phi$	$\cdots$	0
Δ_u	2	$2\cos 2\phi$	$\cdots$	0	-2	$-2\cos 2\phi$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

$\lvert m_1 \rvert$	$\lvert m_2 \rvert$	$\lvert m_1 - m_2 \rvert$	$\lvert m_3 \rvert$	$\lvert m_4 \rvert$	$[m_1 m_2 m_3 m_4]$
0	0	0	0	0	[0 0 0 0]
1	0	1	1	0	[1 0 1 0]
1	1	0	0	0	[1 1 0 0]
			1	1	[1 1 1 1]
		2	1	1	[1 1 1 1]
2	0	2	1	1	[2 0 1 1]
			2	0	[2 0 2 0]
2	1	1	1	0	[2 1 1 1]
			2	1	[2 1 2 1]
		3	2	1	[2 1 2 1]
2	2	0	0	0	[2 2 0 0]
			1	1	[2 2 1 1]
			2	2	[2 2 2 2]
		4	2	2	[2 2 2 2]

Figure 21. Symmetry blocking of two-electron integrals for diatomic molecules.

To illustrate the exploitation of point group symmetry in the four-index transformation stage of calculations on polyatomic molecules, we consider the water molecule at its equilibrium geometry. The relevant molecular point symmetry group is C_{2v}, which has elements E, $C_2(z)$, $\sigma_v(xz)$, and $\sigma_v'(yz)$. The molecule will be taken to be in the xz plane with the oxygen nucleus at the origin and the HOH bond angle bisected by the z axis. E is the identity operator, $C_2(z)$ a rotation by 180° about the z axis, $\sigma_v(xz)$ a reflection in the xz plane, and $\sigma_v'(yz)$ a reflection in the yz plane. The character table for this point group is shown in Table 6.

A general approach to the use of symmetry in four-index transformations is based on carrying out the process in two phases. In phase I, we

Table 6. Character Table for the C_{2v} Point Symmetry Group

C_{2v}	E	$C_2(z)$	$\sigma_v(xz)$	$\sigma_v'(yz)$
A_1	+1	+1	+1	+1
A_2	+1	+1	−1	−1
B_1	+1	−1	+1	−1
B_2	+1	−1	−1	+1

transform from the primitive basis functions to primitive symmetry functions. These primitive symmetry functions are the simplest possible linear combination of basis functions that form symmetry-adapted sets. The basis set is divided into the maximum possible number of disjoint subsets such that each subset contains functions that transform among themselves under the symmetry operations of the point symmetry group. Each primitive symmetry function is a linear combination of functions from one subset. As an example, suppose that we wish to perform a calculation on the water molecule using the following basis set:

$$1s_O 2s_O 2p_{xO} 2p_{yO} 2p_{zO} 1s_{H1} 1s_{H2} 2s_{H1} 2s_{H2} \tag{72}$$

Here the subscript O designates a function centered on the oxygen nucleus and H1 and H2 denote functions centered on the hydrogen nuclei. It can be seen by inspection that the following disjoint subsets of functions that transform among themselves under the operations of the point symmetry group can be devised:

$$(1s_O)(2s_O)(2p_{xO})(2p_{yO})(2p_{zO})(1s_{H1}1s_{H2})(2s_{H1}2s_{H2}) \tag{73}$$

and from these the following primitive symmetry functions can be constructed:

$$\begin{aligned} &1s_O,\ 2s_O,\ 2p_{xO},\ 2p_{yO},\ 2p_{zO},\ 1s_{H1}+1s_{H2},\ 1s_{H1}-1s_{H2},\\ &2s_{H1}+2s_{H2},\ 2s_{H1}-2s_{H2} \end{aligned} \tag{74}$$

An efficient approach is to generate the integrals over the primitive symmetry functions directly and thus obtain substantial economies in the self-consistent-field iterations. The different symmetry block of two-electron integrals that can arise are summarized in Figure 22.

Phase II of the exploitation of symmetry in the four-index transformation algorithm is the transformation from primitive symmetry functions to the molecular orbitals. The integrals over primitive symmetry functions are symmetry blocked and so the transformation is carried out block by block. Each block transformation is an n^5 process, where n is the maximum number of functions of any one symmetry species. For our water molecule example let there be $m(a_1)$, $m(a_2)$, $m(b_1)$, and $m(b_2)$ primitive symmetry functions associated with each of the irreducible representations. The number of nonredundant integrals in each of the blocks of Figure 22 is then

$$[a_1a_1|a_1a_1] \qquad M_1 = m(a_1)[m(a_1)+1][m(a_1)^2+m(a_1)+2]/8 \tag{75a}$$

$$[a_2a_2|a_2a_2] \qquad M_2 = m(a_2)[m(a_2)+1][m(a_2)^2+m(a_2)+2]/8 \tag{75b}$$

$$[b_1b_1|b_1b_1] \qquad M_3 = m(b_1)[m(b_1)+1][m(b_1)^2+m(b_1)+2]/8 \tag{75c}$$

ij \ kl	a_1a_1	a_1a_2	a_1b_1	a_1b_2	a_2a_2	a_2b_1	a_2b_2	b_1b_1	b_1b_2	b_2b_2
a_1a_1	X				X			X		X
a_1a_2		X								
a_1b_1			X							
a_1b_2				X						
a_2a_2					X			X		X
a_2b_1						X				
a_2b_2							X			
b_1b_1								X		X
b_1b_2									X	
b_2b_2										X

Figure 22. Symmetry blocking of two-electron integrals for the water molecule with C_{2v} symmetry. × denotes a block of integrals that are not zero for symmetry reasons.

$$[b_2b_2|b_2b_2] \qquad M_4 = m(b_2)[m(b_2)+1][m(b_2)^2+m(b_2)+2]/8 \tag{75d}$$

$$[a_1a_2|a_1a_2] \qquad M_5 = m(a_1)\,m(a_2)[m(a_1)\,m(a_2)+1]/2 \tag{75e}$$

$$[a_1b_1|a_1b_1] \qquad M_6 = m(a_1)\,m(b_1)[m(a_1)\,m(b_1)+1]/2 \tag{75f}$$

$$[a_1b_2|a_1b_2] \qquad M_7 = m(a_1)\,m(b_2)[m(a_1)\,m(b_2)+1]/2 \tag{75g}$$

$$[a_2b_1|a_2b_1] \qquad M_8 = m(a_2)\,m(b_1)[m(a_2)\,m(b_1)+1]/2 \tag{75h}$$

$$[a_2b_2|a_2b_2] \qquad M_9 = m(a_2)\,m(b_2)[m(a_2)\,m(b_2)+1]/2 \tag{75i}$$

$$[b_1b_2|b_1b_2] \qquad M_{10} = m(b_1)\,m(b_2)[m(b_1)\,m(b_2)+1]/2 \tag{75j}$$

$$[a_1a_1|a_2a_2] \qquad M_{11} = m(a_1)\,m(a_2)[m(a_1)+1][m(a_2)+1]/4 \tag{75k}$$

$$[a_1a_1|b_1b_1] \qquad M_{12} = m(a_1)\,m(b_1)[m(a_1)+1][m(b_1)+1]/4 \tag{75l}$$

$$[a_1a_1|b_2b_2] \qquad M_{13} = m(a_1)\,m(b_2)[m(a_1)+1][m(b_2)+1]/4 \tag{75m}$$

$$[a_2a_2|b_1b_1] \qquad M_{14} = m(a_2)\,m(b_1)[m(a_2)+1][m(b_1)+1]/4 \tag{75n}$$

$$[a_2a_2|b_2b_2] \qquad M_{15} = m(a_2)\,m(b_2)[m(a_2)+1][m(b_2)+1]/4 \tag{75o}$$

$$[b_1b_1|b_2b_2] \qquad M_{16} = m(b_1)\,m(b_2)[m(b_1)+1][m(b_2)+1]/4 \tag{75p}$$

We let

$$M = M_1 + M_2 + \cdots + M_{16} \tag{76}$$

Table 7. Number of Nonzero Two-Electron Integrals before and after Symmetry Blocking for the Water Molecule with C_{2v} Symmetry[a]

n	$m(a_1)$	$m(a_2)$	$m(b_1)$	$m(b_2)$	N	M	R
8	2	2	2	2	666	138	20.7%
26	10	8	6	2	61776	13340	21.6%
26	26	0	0	0	61776	61766	100.0%
100	25	25	25	25	12753775	2059400	15.8%
200	50	50	50	50	202015050	31765050	15.7%
1000	250	250	250	250	125250375250	19594125250	15.6%

[a] n is the total number of basis functions. $m(x)$ is the total number of symmetry-adapted basis functions of symmetry x. N is the number of nonredundant integrals if symmetry is not exploited $[=n(n+1)(n^2+n+2)/8]$. M is the number of nonredundant integrals when symmetry is exploited. R is the ratio M/N.

This is the total number of nonredundant integrals that have to be considered when symmetry is recognized. The advantages resulting from the exploitation of symmetry are underlined in Table 7, where the number of integrals that have to be considered when symmetry is recognized is compared with the number that arise when symmetry is ignored. In this table, R denotes the ratio of the number of nonredundant integrals that arise when symmetry is used, M, to the number that occur when symmetry is ignored, N. It can be seen that R is smallest for a given number of basis functions, n, when the number of functions associated with each symmetry type is equal. Furthermore, the ratio R decreases as n increases if the ratio $m(a_1):m(a_2):m(b_1):m(b_2)$ is maintained.

We conclude this section by noting that many of the advantages resulting from the exploitation of the point group symmetry of a molecule are lost once a nonsymmetrical nuclear configuration is considered—for example, in the calculation of a potential energy surface.

5. Partial Four-Index Transformations

5.1. The Use of Partial Transformations

In some applications it is not necessary to perform a full four-index transformation. For example, in applications of many-body perturbation theory using a Moeller–Plesset zero-order Hamiltonian the second-order energy of a closed-shell system is given by[22,23]

$$E_2=\frac{1}{4}\sum_{i,j}\sum_{a,b}\frac{\langle ij|\dfrac{1-(12)}{r_{12}}|ab\rangle\langle ab|\dfrac{1-(12)}{r_{12}}|ij\rangle}{\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b} \tag{77}$$

where the ε_p are orbital energies and the indices label spin orbitals. i and j label occupied spin orbitals while a and b label unoccupied spin orbitals. (12) is the permutation operator that interchanges the labels "electron 1" and "electron 2." The second-order energy often accounts for the vast majority of the correlation energy of a closed-shell system for which a single determinant forms an appropriate reference function. For example, over 99% of the correlation energy, defined with respect to a matrix Hartree–Fock reference function, can be accounted for in the case of the ground state of the argon atom by means of second-order many-body perturbation theory.[24] Since the only integrals involved in the second-order energy expression are those of the type $[ia|jb]$ there is clearly a need to devise an efficient scheme for performing the partial transformation necessary to obtain this type of integral over molecular orbitals.

In this section, we consider two approaches to the partial transformation problem. The method proposed in 1978 by Pendergast and Hayes is described in Section 5.2. The more recent and more efficient algorithm described by Saunders and van Lenthe is considered in Section 5.3.

5.2. The Pendergast–Hayes Method

Starting from the list of integrals over basis functions $[pq|rs]$, Pendergast and Hayes[7] devised a scheme for performing a partial four-index transformation to obtain integrals over molecular orbitals of the type $\langle ij|ab\rangle = [ia|jb]$. As for the full four-index transformation, they suggested that this could be efficiently achieved by means of four single index transformations of the form

$$[iq|rs] = \sum_{p=1}^{n} t_{pi}[pq|rs] \tag{78a}$$

$$[iq|js] = \sum_{r=1}^{n} t_{rj}[iq|rs] \tag{78b}$$

$$[ia|js] = \sum_{q=1}^{n} t_{qa}[iq|js] \tag{78c}$$

$$[ia|jb] = \sum_{r=1}^{n} t_{sb}[ia|js] \tag{78d}$$

It can be shown that for a closed-shell system described by m doubly occupied orbitals the number of operations required to carry out this transformation is

$$\tfrac{1}{2}[mn^4 + 3n^3(n^2+m) + n^2(m - 3m^3 - m^2) + n(m^4 - m^2)] \tag{79}$$

where n is the number of basis functions.

A flow diagram for the partial transformation scheme of Pendergast and Hayes is shown in Figure 23. In the first step, the integral list is expanded to include all integrals satisfying

$$1 \leqslant p \leqslant n \qquad \text{and} \qquad 1 \leqslant r \leqslant n$$

for every

$$1 \leqslant q(q-1)/2 + s \leqslant n(n-1)/2 \tag{80}$$

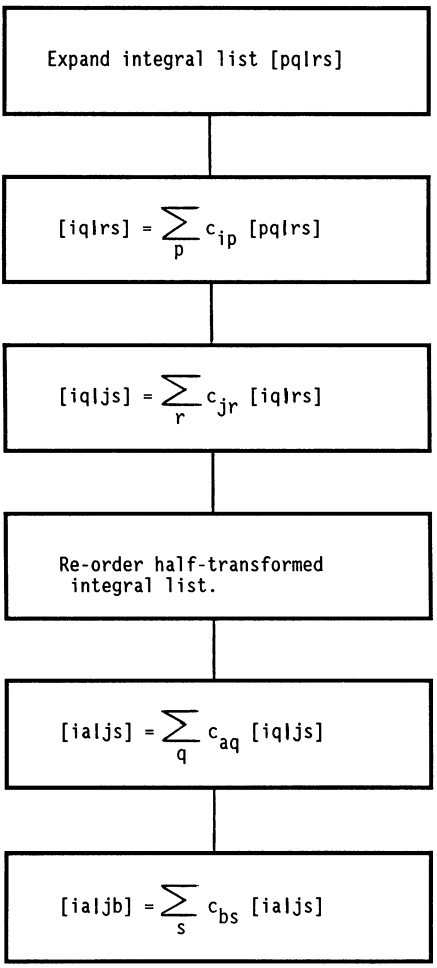

Figure 23. Flow diagram for the partial transformation algorithm of Pendergast and Hayes.

In the fourth step, the integrals are reordered so as to satisfy the condition

$$1 \leqslant q \leqslant n$$

for every ijs $1 \leqslant s \leqslant n$

$$1 \leqslant i(i-1)/2 + j \leqslant m(m-1)/2 \tag{81}$$

5.3. The Saunders–van Lenthe Method

Saunders and van Lenthe[6] suggested an improved partial transformation algorithm. They consider two types of transformed integrals: (i) $[ia|jb]$, which are termed exchange integrals, and (ii) $[ij|\mu\nu]$, where μ and ν label any active molecular orbital. These are termed Coulomb integrals. The algorithm for performing these partial transformations consists of three distinct parts. These are shown in Figures 24, 25, and 26.

The first stage of the partial transformation is shown in Figure 24. This produces the intermediates K and N, which are used to compute the exchange and Coulomb integrals, respectively. The number of floating-point operations involved in this phase of the transformation is

$$\tfrac{3}{8}mn^4 + \tfrac{1}{2}m^2n^3 \tag{82}$$

The algorithm for the transformation of exchange integrals is given in Figure 25. The supermatrix symmetry of the integrals is exploited in this phase of the calculation. The number of floating-point operations involved in this phase of the transformation is

$$\tfrac{5}{3}m^2n^3 \tag{83}$$

The Coulomb integrals are transformed according to the algorithm given in Figure 26. This part of the transformation takes advantage of the overlap distribution symmetry of the integrals. It can be shown that the number of floating-point operations involved is

$$\tfrac{3}{4}m^2n^3 - \tfrac{5}{12}m^3n + \tfrac{1}{12}m^4n \tag{84}$$

In Table 8, we compare the number of multiplications required to perform the partial transformation that is required to obtain the exchange integrals needed to perform a second-order many-body perturbation theory calculation with the number of multiplications that arise in the full transformation.

```
for p:=1 to n step 1
    for q:=1 to p step 1
        for r:=1 to p step 1
            if p=r then smax:=q else smax:=r
            for s:=1 to smax step 1
                for i:=1 to m step 1

                H(pqri) = Σ_s G(pqrs) Q(si)

                next i
            next s
        next r
        for r:=1 to p step 1
            if p=r then smax:=q else smax:=r
            for s:=1 to smax step 1
                for i:=1 to m step 1

                I(pqsi) = Σ_r G(pqrs) Q(ri)

                next i
            next s
        next r
        for r:=p to n step 1
            if p=r then smin:=q else smin:=1
            for s:=smin to r step 1
                for i:=1 to m step 1

                J(pqri) = Σ_s G(pqrs) Q(si)

                next i
            next s
        next r
        for r:=1 to p step 1
            for i:=1 to m step 1

            K(pqri) = J(pqri) + I(pqri)

            next i
        next r
        for r:=1 to n step 1
            for i:=1 to m step 1

            L(pqri) = H(pqri) + J(pqri)

            next i
        next r
        for r:=1 to n step 1
            for i:=1 to m
                for j:=1 to m step 1

                M(pqji) = Σ_r L(pqri) Q(rj)

                next j
            next i
        next r
        for i:=1 to m step 1
            for j:=1 to i step 1

            N(pqji) = M(pqji) + M(pqij)

                    next j
                next i
            next q
        next p
```

Figure 24. First stage of the partial transformation.

```
for r:=1 to n step 1
    for i:=1 to m step 1
        for p:=r to n step 1
            for q:=1 to p step 1
                for j:=1 to m step 1

                A(pjri) = ∑_q K(pqri) Q(qj)

                next j
            next q
        next p
        for p:=r to n step 1
            for q:=1 to p step 1
                for j:=1 to m step 1

                B(qjri) = ∑_p K(pqri) Q(pi)

                next j
            next q
        next p
        for p:=1 to n step 1
            for j:=1 to m step 1

            C(pjri) = A(pjri) + B(pjri)

            next j
        next p
    next i
next r
for i:=1 to m step 1
    for j:=1 to i step 1
        for p:=1 to n step 1
            for r:=1 to n step 1

            D(pjri) = C(pjri) + C(ripj)

            next r
        next p
        for p:=1 to n step 1
            for r:=1 to n step 1
                for a:=m+1 to mm step 1

                E(pjai) = ∑_r D(pjri) q(ra)

                next a
            next r
        next p
        for p:=1 to n step 1
            for a:=m+1 to mm step 1
                for b:=m+1 to mm

                [bjai] = ∑_p E(pjai) Q(pb)

                next b
            next a
        next p
    next j
next i
```

Figure 25. Transformation of exchange integrals.

```
for j:=1 to m step 1
    for i:=j to m step 1
        for p:=1 to n step 1
            for q:=1 to p step 1
                for r:=j to m step 1

                R(prji) = Σ_q N(pqji) Q(qr)


                next r
            next q
        next p
        for p:=1 to n step 1
            for r:=j to m step 1
                if j=r then smin:=i else smin:=j
                for s:=smin to m step 1

                S(srji) = Σ_p R(prji) Q(ps)


                next s
            next r
        next p
        for r:=j to m step 1
            if j=r then smin:=i else smin:=r
            for s:=smin to m step 1

            [rsji] = S(srji) + S(rsji)

            next s
        next r
    next i
next j
```

Figure 26. Transformation of Coulomb integrals.

Table 8. Comparison of the Number of Multiplications Involved in a Partial Transformation to Obtain Integrals Required for Second-Order Many-Body Perturbation Theory with the Number of Multiplications Involved in a Full Transformation

n	m	Partial	Full
20	10	633333	3333333
50	25	61848958	325520833
50	30	85875000	325520833

n is the total number of basis functions, m is the number of doubly occupied orbitals.

6. Approximate Four-Index Transformations

6.1. Approximation Schemes

Because the four-index transformation can be a particularly time-consuming stage of a quantum chemical calculation, the development of approximation schemes has attracted some attention.

Perhaps the most widely employed approximation scheme is the straightforward truncation of the active space; that is, only a subset of the orbitals is explicitly considered both in the four-index transformation and in the subsequent treatment of electron correlation effects. Frequently, the occupied core orbitals are not included in the active space since core correlation effects can often be assumed to be unimportant in the description of chemical phenomena. Unoccupied orbitals associated with high orbital energies are often excluded. In electron correlation energy calculations, these highly excited orbitals only concern the description of core correlation effects. Unfortunately, the exclusion of the core orbitals from the active space also leads to the neglect of core-valence correlation effects, and, although these effects are usually not as important as valence correlation effects in the study of chemical phenomena, they can be quite significant. In the remaining parts of this section, we describe two schemes for the controlled approximation of the four-index transformation. In Section 6.2, we describe a technique based on the Cholesky decomposition of the two-electron supermatrix. Simplifications in the four-index transformation resulting from the use of universal basis sets of even-tempered basis functions[1] are described in Section 6.3.

6.2. Cholesky Decomposition

Beebe and Linderberg[8] have suggested that approximate linear dependence among the columns of the two-electron integral supermatrix together with the fact that the Coulomb operator is positive definite can be exploited in the four-index transformation stage of a quantum chemical calculation. They have demonstrated that the number of two-electron integrals that have to be calculated in order to achieve a chosen accuracy can be significantly reduced. The two-electron integrals can be arranged in the form of a Hermitian positive definite matrix with rows and columns labeled by products of two orbitals

$$(ij) = [i(i-1)]/2 + j \qquad (i \geqslant j) \tag{85}$$

This supermatrix, which will be denoted by V, can be subjected to a

Cholesky decomposition and thus written as the product of a lower triangular matrix L and its adjoint

$$\mathsf{V} = \mathsf{L}\mathsf{L}^{\dagger} \tag{86}$$

Such a decomposition always exists since V is positive definite.

A very stable algorithm can be devise for the construction of L. For $(ij) = 1, 2, \ldots, [n(n+1)]/2$, where (ij) denotes a compound index, we have

$$L_{(ij),(ij)} = \left[V_{(ij),(ij)} - \sum_{(kl)=1}^{(ij)-1} L_{(ij),(kl)} \right]^{1/2} \tag{87}$$

and

$$L_{(mn),(kl)} = \left[V_{(mn),(ij)} - \sum_{(kl)=1}^{(ij)-1} L_{(mn),(kl)} L_{(ij),(kl)} \right] \Big/ L_{(ij),(ij)}$$
$$(mn) = [j(j+1)]/2, \ldots, [n(n+1)]/2 \tag{88}$$

The summations are omitted when the upper index is zero.

Beebe and Linderberg proposed the following algorithm for the construction of the lower triangular matrix L:

(1) Obtain all diagonal elements of V, that is $\mathsf{V}_{(pq),(pq)}$, and arrange them into nonincreasing order. A record of the original order should be kept.

(2) Take the largest diagonal element, $\mathsf{V}_{1,1}$, and set

$$L_{1,1} := V_{1,1}^{1/2} \tag{89}$$

(3) Obtain the entire column of two-electron integrals $V_{(ij),1}$, $(ij) = 1, 2, \ldots, [n(n+1)]/2$.

(4) Put

$$L_{(ij),1} := V_{(ij),1}, \qquad (ij) = 2, 3, \ldots, [n(n+1)]/2 \tag{90}$$

(5) Update the diagonal elements

$$V_{(ij),(ij)} := V_{(ij),(ij)} - L_{(ij),1} L_{(ij),1}, \qquad (ij) = 2, 3, \ldots, [n(n+1)]/2 \tag{91}$$

It should be noted that $V_{1,1}$ is now set to zero.

Now for each value of $(ij) = 2, 3, \ldots, [n(n+1)]/2$ we consider in turn the largest remaining diagonal element $V_{(ij),(ij)}$.

(6) Put

$$L_{(ij),(ij)} := V_{(ij),(ij)}^{1/2} \tag{92}$$

(7) Obtain the partial column $V_{(mn),(ij)}$, $(mn) = [j(j+1)]/2,..., [n(n+1)]/2$.

(8) For $(mn) = [j(j+1)]/2,..., [n(n+1)]/2$ put

$$L_{(mn),(ij)} := \left(V_{(mn),(ij)} - \sum_{(kl)=1}^{(ij)-1} L_{(mn),(kl)} L_{(ij),(mn)} \right) \Big/ L_{(ij),(ij)} \tag{93}$$

(9) Update the diagonal elements according to

$$V_{(mn),(mn)} := V_{(mn),(mn)} - L^2_{(mn),(ij)}, \qquad (mn) = [j(j+1)]/2,..., [n(n+1)]/22 \tag{94}$$

(10) Stop if all remaining updated diagonal elements fall below some specified tolerance, δ.

(11) Go to step (6) and consider the next value of (ij).

Let v be the value of (ij) for which the algorithm stops at step (10). If it is set to the accuracy of the original two-electron integrals then no information of numerical significance has been lost and v is the effective numerical rank of the two-electron integral matrix V. The entire supermatrix V may be regenerated from the truncated lower triangular matrix L by means of

$$V_{(ij),(kl)} = \sum_{(mn)=1}^{M} L_{(ij)(mn)} L_{(kl)(mn)} \tag{95}$$

where

$$M = \min((ij), (kl), v) \tag{96}$$

In practice, the numerical rank, v, of the two-electron integral matrix in a typical quantum chemical calculation is found to be considerably less than its maximum possible value. The approximate four-index transformation scheme of Beebe and Linderberg is clearly useful whenever

$$v \ll [n(n+1)]/2 \tag{97}$$

Now consider a change of orbital basis

$$\phi_p = \sum_i \chi_i C_{pi}$$

The corresponding four-index transformation of the two-electron integrals may be written

$$[pq|rs] = \sum_i \sum_j \sum_k \sum_l C^*_{pi} C^*_{rk} [ij|kl] C_{qj} C_{sl} \tag{98}$$

where $[ij|kl] \equiv V_{(ij),(kl)}$. Substituting the decomposition given in equation (95) yields

$$[pq|rs] = \sum_{(mn)=1}^{M} \sum_i \sum_j \sum_k \sum_l C^*_{pi} C^*_{rk} L_{(ij),(mn)} L_{(kl),(mn)} C_{qj} C_{sl} \tag{99}$$

Putting

$$L'_{(pq),(mn)} = \sum_i \sum_j C^*_{pi} L_{(ij),(mn)} C_{qj} \tag{100}$$

and likewise

$$L'_{(rs),(mn)} = \sum_k \sum_l C^*_{rk} L_{(kl),(mn)} C_{sl} \tag{101}$$

which gives immediately the result

$$[pq|rs] = \sum_{(mn)=1}^{M} L'_{(pq),(mn)} L'_{(rs),(mn)} \tag{102}$$

The time-consuming four-index two-electron integral transformation is reduced to two-index transformations of a limited set of arrays $L_{(pq),(mn)}$.

Crucial to the success of the method of Beebe and Linderberg is the expectation that v is very much less than its maximum possible value in actual calculations. The number of operations, that is, additions, multiplications, and array indexing, reqired to compute the L matrices is $\sim \frac{1}{2}[n(n+1)]\,\frac{1}{2}v^2 - \frac{1}{3}v^3$. The maximum possible value of v is $= [n(n+1)]/2$ giving a maximum of $\sim n^6/48$ operations. A more probable value of v is n, according to the work of Beebe and Linderberg. This gives $\sim n^4/4$ operations. Transformation of a single L matrix requires $\sim n^3/2$ operations and thus the transformation of the entire set of L matrices requires at worst $\sim n^5/4$ but more probably $\sim n^4/2$ operations. Multiplication of the L matrices to obtain the transformed integrals requires $vn^4/8$ operations, giving a probable value of $\sim n^5/8$ or, at worst, $\sim n^6/16$. Beebe and Linderberg[8] have demonstrated the computational tractability of the method.

6.3. Universal Even-Tempered Basis Sets

Following the introduction of powerful techniques based on the linked diagram theorem of many-body perturbation theory to describe electron correlation effects in atoms and molecules within the algebraic approximation (for recent reviews see, for example, Refs. 22 and 23), it quickly became apparent that the dominant source of error in most con-

temporary electronic structure calculations is associated with basis set truncation. One of the most successful techniques for the reduction of the basis set truncation error is the universal basis set (see Ref. 1 for a recent review) and, in particular, the universal basis set of even-tempered basis functions. Calculations in which systematic sequences of even-tempered basis sets[1] are employed can often match the accuracy achieved in self-consistent-field studies of atoms[25] and diatomic molecules[26] using a numerical approach. When correlation effects are considered, the basis set expansion technique can be more accurate than numerical methods because of the difficulties that arise in the integration over the continuum in sum-over-states perturbation expressions.[27] Many-body perturbation theory calculations within the algebraic approximation have now been extended to include relativistic effects.[28]

A detailed overview of the use of universal basis sets and systematic sequences of such basis sets has been given elsewhere.[1,2] Here we discuss those aspects of the use of such basis sets that are relevant to the four-index transformation stage of an electronic structure calculation.

An even-tempered basis set of exponential-type functions or Gaussian-type functions consists of a pure exponential function of pure Gaussian function multiplied by a solid spherical harmonic. The orbital exponents, ζ_k, are chosen to form a geometric sequence

$$\zeta_k = \alpha\beta^k, \qquad k = 1, 2, \ldots, N \tag{103}$$

The universal basis set concept involves the use of the same basis set for a range of nuclear charges and molecular environments.[29] (Clementi and his co-workers[29] have termed these geometric basis sets, see also Ref. 30.) In a systematic sequence of even-tempered basis sets the parameters α and β are taken to be functions of N, the number of basis functions, in such a way that the basis set can approach completeness as the number of functions approaches infinity.

Because universal basis sets are used for a range of atoms and molecules it is clearly worthwhile processing the integrals to ensure that the four-index transformation is performed as efficiently as possible. It would clearly be possible to remove from the list of integrals over the basis function all integrals that have a magnitude below some threshold. Furthermore, a Cholesky decomposition of the type advocated by Beebe and Linderberg[8] can be performed once and for all. It has been shown that the Cholesky decomposition can be effectively exploited in the four-index transformation using universal basis sets.[31]

7. Combined Transformation and Correlation Calculations

7.1. Perturbation Theory

In some approaches to the problem of describing electron correlation effects in atoms and molecules algorithms have been devised that allow the four-index transformation of the two-electron integrals and the calculation of the correlation energy to be carried out simultaneously. Pople and his co-workers[32] demonstrated how such an approach could be incororated into many-body perturbation theory calculations using a Moeller–Plesset reference wave function and Hamiltonian.

The correlation energy of a closed-shell atom or molecule may be described by the many-body perturbation expansion in second and higher orders.[2] The diagrammatic representation of this expansion through third order is shown in Figure 27. There are one second-order diagram and three third-order diagrams, which differ in the nature of the central interaction line that they contain. The algebraic expressions corresponding to the diagrams shown in Figure 27 are

$$E_2 = \frac{1}{4}\sum_{ij}\sum_{ab}\frac{\langle ij|\dfrac{1-(12)}{r_{12}}|ab\rangle\langle ab|\dfrac{1-(12)}{r_{12}}|ij\rangle}{\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b} \tag{104a}$$

$$E_3(h\text{–}h) = \frac{1}{8}\sum_{ijkl}\sum_{ab}\frac{\langle ij|\dfrac{1-(12)}{r_{12}}|ab\rangle\langle kl|\dfrac{1-(12)}{r_{12}}|ij\rangle\langle ab|\dfrac{1-(12)}{r_{12}}|kl\rangle}{(\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b)(\varepsilon_k+\varepsilon_l-\varepsilon_a-\varepsilon_b)} \tag{104b}$$

$$E_3(h\text{–}p) = \sum_{i,k}\sum_{abc}\frac{\langle ij|\dfrac{1-(12)}{r_{12}}|ab\rangle\langle ak|\dfrac{1-(12)}{r_{12}}|ic\rangle\langle bc|\dfrac{1-(12)}{r_{12}}|jk\rangle}{(\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b)(\varepsilon_j+\varepsilon_k-\varepsilon_b-\varepsilon_c)} \tag{104c}$$

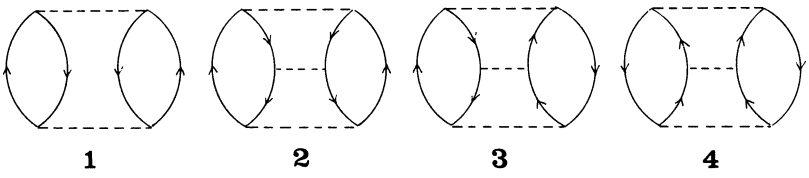

Figure 27. Diagrammatic many-body perturbation theory expansion for the correlation energy of a closed-shell system through third order.

and

$$E_3(p\text{–}p)=\frac{1}{8}\sum_{ij}\sum_{abcd}\frac{\langle ij|\dfrac{1-(12)}{r_{12}}|ab\rangle\langle ab|\dfrac{1-(12)}{r_{12}}|cd\rangle\langle cd|\dfrac{1-(12)}{r_{12}}|ij\rangle}{(\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b)(\varepsilon_i+\varepsilon_j-\varepsilon_c-\varepsilon_d)} \tag{104d}$$

where $i, j, k,...$ are used to denote occupied spin orbitals and $a, b, c,...$ denote unoccupied spin orbitals. The ε's are orbital energies. Computationally, the most demanding of the terms given above is that corresponding to the third-order diagram involving a particle—particle interaction. The evaluation of the expression corresponding to this diagram necessitates summing over indices corresponding to four unoccupied spin orbitals. Pople and his co-workers demonstrated that the four-index transformation and the evaluation of $E_3(p\text{–}p)$ can be usefully combined.

After carrying out the integration over spin coordinates, the energy component associated with $E_3(p\text{–}p)$ may be written[2,22]

$$\begin{aligned}E_3(p\text{–}p)=\frac{1}{8}\sum_{IJ}\sum_{ABCD} & 2\{([IA|JB]-[IB|JA])([AC|BD]-[AD|BC])\\ & \times([CI|DJ]-[CJ|DI])+[IA|JB][AC|BD][CI|DJ]\\ & +[IA|JB][AD|BC][CJ|DI]+[IB|JA][AC|BD][CJ|DI]\\ & +[IB|JA][AD|BC][CI|DJ]\}\\ & \times[(\varepsilon_I+\varepsilon_J-\varepsilon_A-\varepsilon_B)(\varepsilon_I+\varepsilon_J-\varepsilon_C-\varepsilon_D)]^{-1}\end{aligned} \tag{105}$$

where upper case letters are employed for orbitals. Now the four-index transformation required for the integrals $[AC|BD]$ may be written

$$[AC|BD]=\sum_{P,Q,R,S}T^{\dagger}_{AP}T^{\dagger}_{BR}[PQ|RS]\,T_{CQ}T_{DS} \tag{106}$$

Combining equations (105) and (106), we find that the transformation and the evaluation of $E_3(p\text{–}p)$ may be carried out simultaneously, with the summations ordered as follows:

$$\sum_{PQRS}\sum_{ABCD}\sum_{IJ} \tag{107}$$

Of all the integral types that arise in electron correlation energy calculations, those of the type $[AC|BD]$ are, by far, the most numerous. We note, in passing, that a similar approach could be adopted in the determination of $E_3(h\text{–}h)$, but since there is a relatively small number of integrals of the type $[IK|JL]$ there is little to be gained by this approach.

7.2. Coulomb and Exchange Matrices

Some methods for calculating electron correlation energies require simultaneous construction of large numbers of Fock operators, which, in turn, requires the construction of the component Coulomb and exchange matrices.[11] Among such methods are the antisymmetrized product of strongly orthogonal geminals (APSG) model,[33] the multiconfiguration self-consistent-field method (MCSCF),[34] the self-consistent electron pair approach (SCEP),[35] and direct configuration interaction.[6] We note that the methods described in Section 5.3 are suitable for such application.

7.3. Direct Transformation

It has already been indicated that one of the major practical problems in performing *ab initio* molecular structure calculations is the need to store vast numbers of two-electron integrals on peripheral storage devices. In Section 2.5, we have seen that this problem can be alleviated, to some extent, by employing compressed integral lists. However, for the large basis sets that are required for accurate calculations on all but the smallest of molecules, even the use of compressed lists may not be sufficient to completely overcome the integral storage problem. The storage problem becomes particularly severe in calculations that go beyond the Hartree–Fock model and take account of electron correlation effects.

In 1982, Almhof and his co-workers[36] suggested a modification of the usual self-consistent-field algorithm in which the storage of the two-electron integral list is completely avoided. In this scheme the two-electron integrals are reevaluated each time they are required. Of course, additional computer time is required for such an approach, but Almhof and his co-workers demonstrated that this increase is often not unacceptable in self-consistent-field calculations for large molecules.

In this section, we wish to suggest the use of a direct transformation procedure in calculations that take account of electron correlation effects. By this approach we aim to avoid the need to store both the integrals over the basis functions (including the primitive symmetry functions) and the integrals over the molecular orbitals.

Assume that a self-consistent-field calculation has already been performed using the direct self-consistent-field method. We, therefore, have the orbital expansion coefficients and the orbitals energies. Correlation effects are to be treated by many-body perturbation theory or, perhaps, direct configuration interaction or coupled-cluster expansions.[2] Blocks of integrals over molecular orbitals may be obtained by explicitly carrying out the four-index transformation as they are required. Such an approach would be particularly tractable in the case of many-body perturbation

theory calculations, since, unlike direct configuration interaction or coupled-cluster expansions, they are not iterative. In addition, second-order many-body perturbation theory calculations only require a partial four-index transformation of the type discussed in Section 5.

8. The Use of Vector Processing Computers

8.1. Vector Processing Computers

The introduction of vector processing computers, such as the CRAY 1 and the CYBER 205 in the late 1970s and early 1980s, had a most significant impact on quantum chemical calculations in general[37] and on the development of efficient four-index transformation algorithms in particular. Before the arrival of these so-called supercomputers, the four-index transformation had been something of a bottleneck in quantum chemical calculations, particularly when high accuracy was demanded and, therefore, a large basis set had to be employed. However, even early applications of the CRAY 1 vector processing computer to the transformation problem were found to lead to a near maximum rate of computation.[37]

A very simple model of a vector processing computer is displayed in Figure 28 together with a model of a serial machine. A serial computer not only performs floating point operations (addition, multiplication, division) sequentially, but also carries out the various suboperations involved sequentially. Thus if a floating point multiplication requires m suboperations then a serial computer will require m clock cycles to perform the multiplication. This is illustrated in part (a) of Figure 28. On a vector processor the various suboperations are performed concurrently. Thus although a single float point operation still requires m clock cycles, for a vector containing a sufficient number of elements, results can be produced at the rate of one per clock cycle. This is illustrated in part (b) of Figure 28.

8.2. The Four-Index Transformation on Vector Processors

As we have already pointed out in Section 3.3, the four-index transformation may be written as two two-index transformations [see equation (28)]:

$$\begin{aligned} [pq|kl] &= \sum_{r=1}^{n} \sum_{s=1}^{n} T_{kr}^{\dagger} [pq|rs]\, T_{ls}, \qquad p, q, k, l = 1,\ldots, n \\ [ij|kl] &= \sum_{p=1}^{n} \sum_{q=1}^{n} T_{ip}^{\dagger} [pq|kl]\, T_{jq}, \qquad i, j, k, l = 1,\ldots, n \end{aligned} \tag{108}$$

The key to the development of efficient algorithms, therefore, lies in the construction of techniques for matrix algebra and, in particular, for matrix multiplication.[37,38]

The matrix product $\mathsf{C} = \mathsf{AB}$ may be written as

$$C_{ij} = \sum_k A_{ik} B_{kj} \tag{109}$$

and the evaluation of the element C_{ij} of the resultant matrix may, therefore, be regarded as the evaluation of the scalar product of the vector $\mathbf{a}_i$, consisting of the ith row of the matrix A, and the vector $\mathbf{b}_j$, consisting of the jth column of the matrix B. Timing data for the evaluation of the scalar product of two vectors of length n on the Cyber 205 vector processor is shown in Table 9. As n increases, the ratio t/n, where t is the total time required to evaluate the scalar product, decreases to a limiting value. The limiting value is equal to the cycle time of the Cyber 205, 20 nsec. It can be seen from Table 9 that quite long vectors are required on the Cyber 205 before the maximum rate of computation is attained. For the CRAY 1 machine the maximum rate of computation is achieved for somewhat shorter vector lengths.

It has been suggested[5,8] that the Winograd technique[39] for matrix

Table 9. Timing Data for the Scalar Product on the CYBER 205

n	t (msec)	t/n (msec)
100	0.005117	0.000051
200	0.007045	0.000035
300	0.009123	0.000030
400	0.011044	0.000028
500	0.013123	0.000026
600	0.015044	0.000025
700	0.017123	0.000024
800	0.019044	0.000024
900	0.021124	0.000023
1000	0.023076	0.000023
2000	0.043054	0.000022
3000	0.063049	0.000021
4000	0.083134	0.000021
5000	0.103166	0.000021
6000	0.123050	0.000021
7000	0.143268	0.000020
8000	0.163232	0.000020
9000	0.183055	0.000020
10000	0.203317	0.000020

multiplication should be exploited in the four-index transformation stage of an electronic structure calculation. Winograd suggested that the matrix product

$$C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj} \tag{110}$$

be rewritten as

$$C_{ij} = \sum_{k=2,4,\ldots}^{N} (A_{ik-1} + B_{kj})(A_{ik} + B_{k-1\,j}) - \sum_{k=2,4,\ldots}^{N} A_{ik-1} A_{ik} - \sum_{k=2,4,\ldots}^{N} B_{k-1\,j} B_{kj} \tag{111}$$

where N is taken to be even.

This reduces the number of multiplications required from N^3 to $\frac{1}{2}N^3 + N^2$, where N is the dimension of the matrix. Some care needs to be taken to avoid loss of accuracy.[40] However, it should be noted that although the Winograd technique reduces the number of multiplications, it also increases the number of additions and, therefore, on a computer such as the Cyber 205, on which addition and multiplication take identical amounts of time, the Winograd algorithm is actually slower than the conventional approach.

We note that the Winograd technique can be generalized to the case where N is odd as follows:

$$C_{ij} = \sum_{k=2,4,\ldots}^{N-1} (A_{ik-1} + B_{kj})(A_{ik} + B_{k-1\,j}) + A_{iN} B_{Nj} - \sum_{k=2,4,\ldots}^{N-1} A_{ik-1} A_{ik} - \sum_{k=2,4,\ldots}^{N-1} B_{k-1\,j} B_{kj} \tag{112}$$

9. The Use of Parallel Processing Computers

9.1. Parallel Processing Computers

Parallel processing computers promise an even greater impact on computational quantum chemistry[11] and, in particular, on the four-index transformation of the two-electron integrals than did the advent of the vector processing computer. Indeed, a central problem of computational quantum chemistry over the next decade is the development of efficient algorithms that allow for the organization of concurrent computation on a

very large scale. Hockney and Jesshope[41] summarize one important result of experience gained in using vector processing and parallel processing computers to date:

> What has changed with the advent of the parallel computer is the ratio between the performance of a good and a bad computer program. This factor is not likely to exceed a factor of two on a serial machine, whereas factors of ten or more are not uncommon on parallel computers.

The distinction between a vector processing computer and a parallel computer can be understood by comparing Figures 28 and 29. Whereas on a vector processor the various suboperations that comprise a floating point operation are performed simultaneously, giving a maximum rate of computation of one result every clock cycle, the parallel processor performs whole floating point operations concurrently.[11]

In fact, parallel processor architecture can vary enormously. They range from a small number of very powerful processors, such as the ETA[10], which consists of an array of up to eight processors working simultaneously—each processor being a powerful vector processor itself—to machines consisting of a large number of processors, such as the FPS-T series, which consists of 16384 processors, each of which is a transputer (a processor on a single chip).

For systems consisting of a large number of processors not only are the characteristics of the individual processors important but the interconnections between these processors can be critical. A structure having particularly interesting and useful interconnection features is the hypercube.[42] An n-cube consists of 2^n nodes, each associated with a single processor and numbered by an n-bit binary digit between 0 and $2^n - 1$. The processors are interconnected so that there is a link between two processors if and only if their binary representation differs by one and only one bit. For example, in the case $n = 3$ the eight processors can be represented as

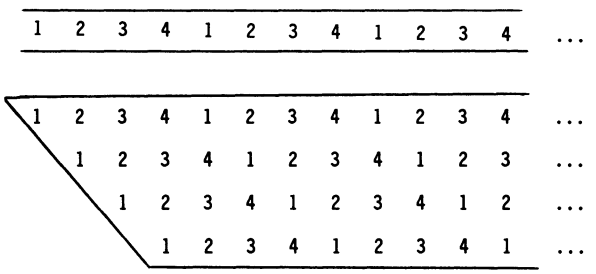

Figure 28. A vector processing computer. A simple model. The upper figure represent a serial computer and the lower figure a vector processor. The digits 1, 2, 3, 4 represent the suboperations involved in a floating point operation.

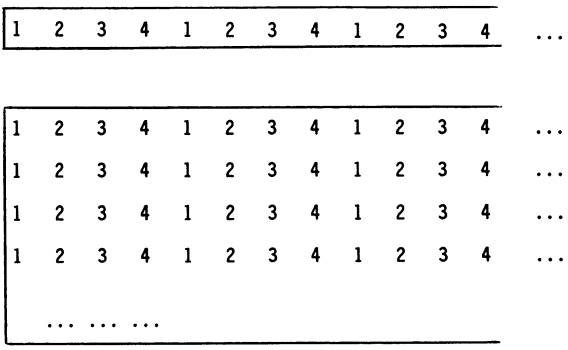

Figure 29. A parallel processing computer. A simple model. The upper figure represents a serial computer and the lower figure a parallel processor. The digits 1, 2, 3, 4 represent the suboperations involved in a floating point operation.

the vertices of a three-dimensional cube. The hypercube architecture contains many classical topologies, such as two-dimensional and three-dimensional meshes. In fact, it contains meshes of arbitrary dimension. It has homogeneity and symmetrical properties since no single processor plays a particular role.

Early programming languages, such as FORTRAN, were designed for serial machines. They are not particularly well suited to parallel architectures. New languages, such as OCCAM,[43] have therefore been developed in order to express the parallelism in a particular problem more succinctly. The four-index transformation, being the step in most quantum chemical calculations most likely to benefit from the exploitation of parallelism, will provide a rigorous test of the applicability of these languages.[11]

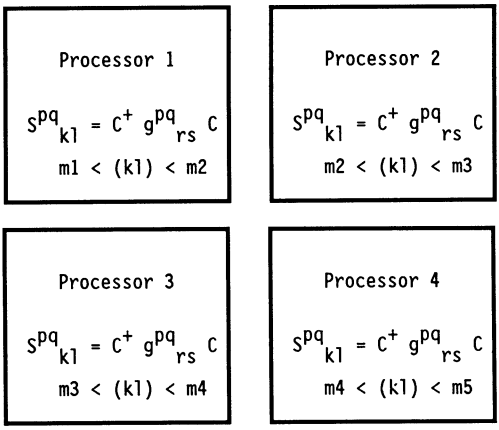

Figure 30. The four-index transformation on parallel processors.

9.2. The Four-Index Transformation on Parallel Processors

Little work has been done on the efficient implementation of the four-index transformation on parallel processing computers. It is to be expected that such machines will have a significant impact on the transformation phase of quantum chemical calculations.

It is, perhaps, unlikely that a single algorithm can be designed for the four-index transformation that will be optimal on all types of parallel processors. The preferred algorithm will probably have a strong dependence on the architecture of the particular machine being used.

As an example of what may become possible with a parallel processor such as ETA[10] having a small number of very powerful processors, we sketch in Figure 30 a subdivision of tasks in the four-index transformation. It is proposed that the four-index transformation be performed as two two-index transformations and that these two-index transformations could be performed in parallel. On an eight processor ETA[10] machine the four-index transformation could then be carried out in one eighth of the time taken on a single processor.

10. Summary and Future Prospects

In this chapter, we have reviewed the methods employed in contemporary atomic and molecular electronic structure calculations to perform efficient four-index transformations of two-electron integrals. This is an essential first step in the calculation of correlation energies. In addition to the algorithms and computational methods discussed in Sections 2 and 3, we have described the exploitation of point group symmetry, the use of partial four-index transformations, approximate transformations, combined transformation and correlation energy calculations, and the use of vector processing and parallel processing computers for four-index transformations.

In looking to the future, it is clear that the development of efficient techniques for performing four-index transformations of two-electron integrals, perhaps more than for any of the other steps in *ab initio* electronic structure calculations, is going to be driven by computer technology. Its dependence on the fifth power of the number of basis functions frequently makes the four-index transformation the most computationally demanding step in electronic structure studies. (Recall that integral evaluation, self-consistent-field calculations, and second-order many-body perturbation theory treatments of correlation effects all depend on the fourth power of the number of basis functions.[11]) However, the four-index transformation is also the step in which the advantages of vector processors

and parallel processors can often be exploited to their fullest extent. The use of partial transformations is clearly of importance in extending size-consistent methods, such as many-body perturbation theory, to large molecules. The integral storage problem could be avoided by using a direct transformation procedure of the type suggested in Section 7.3. One can envisage a direct many-body perturbation theory (or direct direct CI for small molecules) in which integral storage problems, a practical feature of many contemporary large-scale *ab initio* calculations, are almost completely avoided.

References

1. S. Wilson, Basis sets, *Adv. Chem. Phys.* **67**, 439 (1987).
2. S. Wilson, *Electron Correlation in Molecules*, Clarendon Press, Oxford (1984).
3. C. F. Bender, Integral transformations: A bottle neck in molecular quantum mechanical calculations, *J. Comp. Phys.* **9**, 547–554 (1972).
4. I. Shavitt, The method of configuration interaction, in: *Methods in Electronic Structure Theory* (H. F. Schaefer III, ed.), Plenum Press, New York (1977).
5. S. T. Elbert, Four index integral transformations: An $n^{4.?}$ problem? in: *Numerical Algorithms in Chemistry: Algebraic Methods*, Report of NRCC Workshop August 9–11, 1978, pp. 129–141, LBL-8158 (1978).
6. V. R. Saunders and J. H. van Lenthe, The direct CI method. A detailed analysis, *Molec. Phys.* **48**, 923–954 (1983).
7. P. Pendergast and E. F. Hayes, A partial transformation for application to perturbation theory configuration interaction, *J. Comp. Phys.* **26**, 236–242 (1978).
8. N. H. Beebe and J. Linderberg, Simplifications in the generation and transformation of two-electron integrals in molecular calculations, *Int. J. Quantum Chem.* **12**, 683 (1977).
9. O. Steinborn, On the evaluation of exponential (Slater) type integrals, in: *Methods in Computational Molecular Physics* (G. H. F. Diercksen and S. Wilson, eds.), Reidel, Dordrecht (1983).
10. V. R. Saunders, Molecular integrals for Gaussian-type functions, in: *Methods in Computational Molecular Physics* (G. H. F. Diercksen and S. Wilson, eds.), Reidel, Dordrecht (1983).
11. S. Wilson, *Computational Quantum Chemistry*, Clarendon Press, Oxford (1988).
12. F. Billingsley, *Int. J. Quantum Chem.* **12**, 843 (1972).
13. M. Yoshimine, The use of direct access devices in problems requiring the reordering of long lists of data, Report RJ-555 IBM Research Laboratory, San Jose, California, 1969.
14. K. C. Tang and C. Edmiston, More efficient method for the basis transformation of electron interaction integrals, *J. Chem. Phys.* **52**, 997 (1970).
15. R. K. Nesbet, *Rev. Mod. Phys.* **35**, 552 (1963).
16. A. D. McLean, Potential energy surfaces from ab initio computation: Current and projected capabilities of the ALCHEMY computer program, in: *Proceedings of the Conference on Potential Energy Surfaces in Chemistry* (W. A. Lester, Jr., ed.), IBM (1971).
17. M. Yoshimine, in *Energy, Structure and Reactivity* (D. W. Smith and W. B. McRae, eds.), p. 143, Wiley, New York (1973).
18. P. S. Bagus, A. D. McLean, and M. Yoshimine, in *Energy, Structure and Reactivity* (D. W. Smith and W. B. McRae, eds.), p. 130, Wiley, New York (1973).
19. G. H. F. Diercksen, Optimized transformation of four centre integrals, *Theoret. Chim. Acta* **33**, 1 (1974).

20. C. N. M. Pounder, The two-electron integral transformation and two-body density matrix transformation, *Theoret. Chim. Acta* **39**, 247–253 (1975).
21. P. Pendergast and W. H. Fink, A thorough analysis and exposition of the four-index transformation, *J. Comp. Phys.* **14**, 286–300 (1974).
22. S. Wilson, Diagrammatic many-body perturbation theory of atomic and molecular structure, *Comput. Phys. Rep.* **2**, 389–482 (1985).
23. S. Wilson, *The Many-Body Perturbation Theory of Atoms and Molecules*, Adam Hilger, Bristol (1988).
24. B. H. Wells and S. Wilson, Second-order correlation energy of the argon atom using basis sets of gaussian-type functions, *J. Phys. B: At. Mol. Phys.* **19**, 2411 (1986).
25. M. W. Schmidt and K. Ruedenberg, Effective convergence to complete orbital bases and to the atomic Hartree–Fock limit through systematic sequences of Gaussian primitive, *J. Chem. Phys.* **71**, 3951 (1979).
26. B. H. Wells and S. Wilson, On the accuracy of the algebraic approximation for diatomic molecules, *J. Phys. B: At. Mol. Phys.* **18**, L731 (1985).
27. H. M. Quiney, I. P. Grant, and S. Wilson, Diagrammatic perturbation theory: A comparison of numerical methods with basis set expansion techniques for a model problem, *J. Phys. B: At. Mol. Phys.* **18**, 577 (1985).
28. H. M. Quiney, I. P. Grant and S. Wilson, The Dirac equation in the algebraic approximation. III. Diagrammatic perturbation theory calculations for a model problem, *J. Phys. B: At. Mol. Phys.* **18**, 2805 (1985).
29. E. Clementi and G. Corongiu, Geometrical basis sets for molecular computations, IBM Research Report, Poughkeepsie, New York, 1982.
30. S. Huzinaga, Basis sets for molecular calculations, *Comput. Phys. Rep.* **2**, 279 (1985).
31. S. Wilson, Universal basis sets and Cholesky decomposition of the two-electron integral matrix, to be published.
32. J. A. Pople, R. Seeger, and R. Krishnan, Variational configuration interaction methods and comparison with perturbation theory, *Int. J. Quantum Chem.* **S11**, 149 (1977).
33. D. M. Silver, K. Ruedenberg, and E. L. Mehler, Electron correlation and the separated electron pair approximation in diatomic molecules. III, *J. Chem. Phys.* **52**, 1206 (1970).
34. B. O. Roos, The multiconfigurational SCF method, in: *Methods in Computational Molecular Physics* (G. H. F. Diercksen and S. Wilson, eds.), Reidel, Dordrecht (1983).
35. W. Meyer, *J. Chem. Phys.* **64**, 2901 (1975); R. Ahlrichs, *Comput. Phys. Commun.* **17**, 31 (1979).
36. J. Almhof, K. Faegri, and K. Korsell, Principles for a direct SCF approach to LCAO-MO ab initio calculations, *J. Comp. Chem.* **3**, 385 (1982).
37. M. F. Guest and S. Wilson, The use of vector processors in quantum chemistry, in: *Supercomputers in Chemistry* (P. Lykos and I. Shavitt, eds.), American Chemical Society, Washington, D.C. (1981).
38. V. R. Saunders and M. F. Guest, Applications of the Cray 1 for quantum chemical calculations, *Comput. Phys. Commun.* **26**, 389 (1982).
39. S. Winograd, A new algorithm for inner products, *IEEE Trans. Comput.* **17**, 693 (1968).
40. R. P. Brent, Error analysis of algorithms for matrix multiplication and triangular decomposition using Winograd's identity, *Numer. Math.* **16**, 145 (1970).
41. R. W. Hockney and C. R. Jesshope, *Parallel Computers: Architecture, Programming and Algorithms*, Adam Hilger, Bristol (1981).
42. L. N. Bhuyan and D. P. Agrawal, Generalized hypercube and hyperbus structures for a computer network, *IEEE Trans. Comput.* **C-33**, 322 (1984).
43. Inmos, *The* OCCAM *Programming Manual*, Prentice Hall, Englewood Cliffs, New Jersey (1985).