Theoret. Chim. Acta (Berl.) 34, 105—114 (1974)

Configuration Interaction by the Method of Bonded Functions, Some Preliminary Calculations

Geerd H. F. Diercksen
Max-Planck-Institut für Physik und Astrophysik, München, Germany

B. T. Sutcliffe
University of York, York, England

Received December 12, 1973

A resumé of Boys' approach to configuration interaction calculations is presented, and a program suitable to perform such calculations is described in some detail. The results of a preliminary calculation on water, together with some timings are presented.

Key words: Configuration interaction – Bonded functions – Water – Program timings

1. Introduction

If one chooses to attempt approximate solutions of Schrödinger's equation for bound states of atoms and molecules, with the aid of the linear variation theorem, then one begins with the ansatz

$$\psi = \sum_{k=1}^{m} c_k \Phi_k \tag{1}$$

and is eventually faced with solving the secular problem

$$\boldsymbol{H} c = E \boldsymbol{S} c\,. \tag{2}$$

Here $\boldsymbol{H}$ and $\boldsymbol{S}$ are square matrices with elements

$$H_{ij} = \int \Phi_i^* H \Phi_j d\tau \tag{3a}$$

$$S_{ij} = \int \Phi_i^* \Phi_j d\tau \tag{3b}$$

where H is the Schrödinger hamiltonian for the problem

$$H = \sum_i H(i) + \sum_{i>j} H(i,j) \tag{4a}$$

with

$$H(i) = -\frac{1}{2}\Delta_i + \sum_\lambda Z_\lambda / r_{\lambda i} \tag{4b}$$

$$H(ij) = 1/r_{ij}\,. \tag{4c}$$

The eigenvectors c consist of those coefficients in (1) which minimize the energy E.

Nowadays the solution of the eigenvalue problem does not present any particular computational difficulties, but obtaining the matrix elements (3) and (4) still presents a formidable computational problem.

In quantum chemical problems the Φ_k are usually taken to be antisymmetrised products of one particle space and spin functions (*spin orbitals*) and it can be seen at once that with this choice the matrix elements (3) reduce to weighted sums of one- and two-electron (three and six dimensional) integrals. The evaluation of these integrals is again a matter of great computational difficulty, with consequences for the evaluation of the matrix elements to which we shall refer later.

If one chooses the Φ_k as antisymmetrised spin-orbital products, then a still further choice is left open, that of choosing the space parts of the functions (orbitals) as members or not, of an orthogonal set.

If one chooses them to form an orthogonal set then many simplifications appear in the formulae for the matrix elements. However, it has not so far been found possible to evaluate directly the integrals involved over any physically meaningful or useful set of orthogonal orbitals. Generally orthogonal orbitals are constructed as linear combinations of primitive functions, by some means. The primitive functions are chosen for the ease with which integrals between them may be evaluated and also on grounds of physical meaning. Thus before one can actually evaluate the matrix elements in an integral basis one has generally to face the problem of transforming the integrals from the primitive basis to the orthogonal basis. Only recently has this problem been solved in a computationally efficient way, and this has been discussed by one of us (G.H.F.D.) in another paper [1]. In this context it is the custom to refer to the primitive functions as atomic orbitals (AO's) and to the orthogonal functions as molecular orbitals (MO's) because the orthogonal functions were often found as solutions of Roothaan's equations. It is also quite customary to refer to the process of constructing the linear variation functions as *configuration interaction* (CI).

If one does not require the orbitals to form an orthogonal set then one has no transformation problem but the weighting function in the integral sums then involves the evaluation of a rather nasty co-factor expression (see for example [2] pp. 50–51) and it seems likely that the amount of computational effort involved in evaluating the matrix elements here, may well in fact be very similar to that involved in transforming and evaluating in the orthogonal basis. This is, however, as yet an undecided question.

In this communication we shall confine attention to matrix elements in an orthogonal basis, and because of this we shall be able to consider a somewhat more general functional form for the Φ_k than the relatively simple antisymmetrised product. This functional form (which we shall describe in more detail later) we shall, following the usage of Boys, call a *bonded function.* It may be thought as a linear combination of antisymmetrised products, so designed as to be a spin-eigenfunction and to have the required space symmetry properties. Such a functional form has the further advantage that it is easy to generate from any given orbital set all those bonded functions having the same spin-eigenvalue

(corresponding to the different canonical structures of classical valence bond theory (see e.g. [2] p. 67)) so that one may properly consider all allowed spin-coupling schemes in any problem, in an economical way.

Some of the earliest considerations of the problem of generating bonded functions and calculating matrix elements between them, from the standpoint of computational feasibility, are found in the work of McWeeny [3] and of Boys and his co-workers ([4, 5]). Subsequently these approaches were somewhat generalised and extended by McWeeny and Cooper [6] and by Sutcliffe [7] respectively. We shall not concern ourselves here with the problem of generating a suitable set of bonded functions but will regard such a set as given, and concentrate on the computational problems raised by finding the formula for the matrix element between an arbitary pair of bonded functions and of subsequently substituting the values of the integrals into this formula to obtain the required matrix element. We shall call the first part of this process (again following Boys) the *projective reduction* of a matrix element to yield a *symbolic matrix element*, and the last part that of forming the *numerical matrix element*, by resolving the symbolic references.

From a more general point of view the symbolic matrix element (or indeed a complete list of such elements) can be regarded as a special kind of program, according to the execution of which, the numerical value is computed. The program which generates the symbolic matrix elements can then be regarded as a compiler, generating from input, the symbolic matrix element regarded as a program, according to the syntax rules and so on of projective reduction. The formation of numerical matrix elements may then be regarded as interpreting the compiled symbolic matrix element program.

In certain cases, as Roos [8] has shown, it is possible to look at this problem from a different viewpoint. If one restricts the structure of the bonded functions in certain ways, then one can so arrange matters that only a small number of possible types of symbolic matrix elements occur. In this kind of situation instead of resolving the references in the symbolic matrix element to the numerical values of the integrals, it is more effective to use the integral type as a symbolic reference and to resolve this reference to all the possible numerical matrix elements. This latter process is very like the technique used, for example, in the Polyatom [9] and Munich [10] SCF routine for making up the J and K matrices and the HF-matrix, resp., by tagging each two electron integral according to type and processing it as a potential contributer to a number of matrix elements according to the tag carried. While recognising the outstanding suitability of Roos' technique in particular cases (for example the classical case of configuration interaction involving all single and double substitutions in a closed shell) it is difficult to see how it could be made to operate in the general case of arbitary bonded functions. We shall therefore not consider it further here since our interest is precisely in this latter situation.

There have, in fact, been quite a number of earlier attempts to treat the problem in the same broad general way that we are proposing, the classical work being that of Boys and Reeves (see [11]) and work by others arising from that. However, the present state of the art appears to be that still it is not possible to regard the calculation of a general say 5000 configuration wave function, as a

routine affair because the computing times involved remain much too great. That this is the case, is almost certainly due (in part) to the fact that no really effective algorithm has been available for interpreting the compiled symbolic matrix element program. Such an algorithm has now been designed, making use of a reordering procedure for large lists of indexed quantities. The algorithm, which will be described later, has been implemented within the scope and framework of the Munich program system [10] and has been extensively tested and found to perform well[1].

2. Theory

The bonded functions Φ which form the basis of our analysis are defined as follows

$$\phi = \mathscr{A}[\phi_1\phi_2][\phi_3\phi_4]\cdots[\phi_{2p-1}\phi_{2p}][\phi_{2p+1}]\cdots[\phi_n] \tag{5}$$

where the spin coupled pairs are

$$[\phi_i\phi_j] = \phi_i(i)\,\phi_j(j)\,\{\alpha(i)\,\beta(j) - \beta(i)\,\alpha(j)\} \qquad \phi_i \neq \phi_j \tag{5a}$$

$$= \phi_i(i)\,\phi_j(j)\,\alpha(i)\,\beta(j) \qquad \phi_i = \phi_j \tag{5a'}$$

and the unpaired orbital is

$$[\phi_i] = \phi_i(i)\,\alpha(i)\,. \tag{5b}$$

The symbol $\mathscr{A}$ denotes an antisymmetrizing operator that produces a normalized, completely antisymmetric wavefunction. The functions ϕ_i are assumed to be orthonormal. If $\phi_i = \phi_j$, then the orbitals must occur in the same spin coupled pair or the function vanishes.

A bonded function composed of n orbitals of which c are unpaired and containing x identical orbitals spin coupled (identical pairs) may be written as the sum of $2^{(n-c)/2-x}$ determinants. A given set of orbitals ϕ may be bracketed together in a number of different ways. A linear independent set of bonded functions (canonical set) may be formed according to the following rules: (a) In each bonded function identical orbitals must be bracketed together (spin coupled). (b) To the remaining orbitals the remaining left and right brackets must be assigned. These have to be assigned one to each orbital in all possible ways consistent with there being at least one more left bracket to the left of any right bracket than there are right brackets. The brackets are associated by the ordinary laws of algebra and the orbitals assigned to each pair of brackets, spin coupled. The excess of left brackets (if any) represents the uncoupled orbitals.

The total number of determinant product terms in the product of two bonded functions Φ_K and $\Phi_{K'}$ containing x and x' identical pairs, respectively, is $2^{(n-c)-(x+x')}$. These determinant product terms must be enumerated, the required matrix element between the determinants must be found, and all the

[1] After completing this work the authors found that a very similar algorithm had been developed simultaneously by Yoshimine [12]. The relative merit of this and the procedure described here is still an open question.

contributing factors summed to give the final matrix elements between the bonded functions. The matrix elements of the unity and the spinless Hamiltonian operator between the bonded functions Φ_K and $\Phi_{K'}$ are determined to be of the form

$$S_{KK'} = \Gamma \sum_i Q_i \int \phi_i^K(1)\, \phi_i^{K'}(1)\, d\tau_1 \tag{6a}$$

$$\begin{aligned} H_{KK'} = {} & \Gamma \sum_i Q_i \int \phi_i^K(1)\, H(1)\, \phi_i^{K'}(1)\, d\tau_1 \\ & + \sum_{i,j} Q_{ij} q'_{ij} \int \phi_i^K(1)\, \phi_i^{K'}(1)\, H(1,2)\, \phi_j^{K'}(2)\, \phi_j^{K'}(2)\, d\tau_1\, d\tau_2 \\ & + q_{ij} \int \phi_i^K(1)\, \phi_j^{K'}(1)\, H(1,2)\, \phi_j^K(2)\, \phi_i^{K'}(2)\, d\tau_1\, d\tau_2 . \end{aligned} \tag{6b}$$

The coefficients Γ, Q_i, Q_{ij}, q_{ij}, and q'_{ij} are constants which are independent of the form of the orbitals and of the operators $H(1)$ and $H(1,2)$ and depend only on the bracket structure of the bonded functions. The process of reducing the many dimensional integrals $S_{KK'}$ and $H_{KK'}$ to a combination of weighted integrals over one- and two-electron coordinate integrals has been termed projective reduction. This projective reduction has to be performed for each matrix element separately according to the following rules:

Let the orbitals ϕ_i^K that compose Φ_K be written in a line and the orbitals $\phi_i^{K'}$ that compose $\Phi_{K'}$ be written down below them. Now let the orbitals of Φ_K and $\Phi_{K'}$ be rearranged so that

1. identical orbitals appear opposite one another as far as possible,
2. spin coupled pairs are kept adjacent as far as possible.

Rule (1) is applied before rule (2) above, and it will only be in the case where the orbitals of Φ_K differ from $\Phi_{K'}$ that identical orbitals will not appear opposite one another. Rule (1) is applied by associating each orbital in Φ_K with the same orbital in $\Phi_{K'}$ until all identical orbitals have been associated. The nonidentical orbitals are then paired and the resulting diagram rearranged so as to confirm with rule (2). In particular cases the diagram produced is not unique, but all such diagrams can be shown to be equivalent. It should be noticed that the orbital subscripts in Eqs. (6a) and (6b) refer to the orbital order after this re-ordering has been done. The numerical value of the subscript is, of course, of no consequence, it is simply required that ϕ_i^K be opposite $\phi_i^{K'}$ and so on after re-ordering.

Patterns are formed by joining orbitals which have been arranged adjacent to each other according to the above rules by a solid line, and connecting all spin coupled pairs by a dotted line. Any diagram consists generally of two types of patterns. Those which begin and end on an unpaired orbital and those which close back on themselves. The former are referred to as chains, the latter as cycles. The chains are of two types: those which begin in one function Φ_K and end in the other $\Phi_{K'}$, these are called odd chains since they involve an odd number of vertical links; and those which begin in one function and end in the same function are called even chains. It is clear that there must be just as many chains in a diagram as there are unpaired orbitals in a bonded function. If there are even chains then there must be at least two and generally an even number of even chains.

Table 1. Coefficients for two-electron integrals

i	j	p_{ij}	Pattern	q_{ij}
Cycle	Cycle	-1	D	$-\frac{1}{2}$
Cycle	o chain	$+1$	D	$-\frac{1}{2}$
o chain	Cycle	-1	S	$+1$
		$+1$	S	-2
o chain	o chain	-1	D	0
		$+1$	D	-1
		-1	S	$+1$
		$+1$	S	-2
e chain	e chain	-1	D	-1
		$+1$	D	$+1$

It is necessary to have a convention about where chains begin. The first odd chain is taken to begin at the lowest numbered unpaired orbital in the top line of the diagram. The next odd chain starts at the next lowest unpaired orbital and so on. The first even chain is defined like the first odd chain, the second even chain starts from the lowest numbered available unpaired orbital in the bottom line of the diagram and so on.

Inspection shows that if any even chains are present then there must be one spin mismatched for each even chain, between the determinants. By convention this is taken to be at the highest numbered orbital in the chain.

A parity is assigned to each vertical line within a pattern, the lowest numbered line being even, the next odd, the next even, and so on.

Now the patterns can be used to determine the sign of the initial diagram and also to write down the matrix elements between bonded functions in terms of integrals over the orbitals of ϕ^K, and of $\phi^{K'}$.

The results are given by formulas (6) and Table 1. The notation convention adopted is as follows: The parity of a given position ($+1$ or -1) is denoted by p_i. The product $p_i p_j$ is written as p_{ij}. If i and j occur in different patterns this is denoted by D, if they occur in the same pattern this is denoted by the letter S. The function Q_i is zero, if there exist an $r \neq i$, such that $\phi_r^K \neq \phi_r^{K'}$, and it is one otherwise. Similarly the function Q_{ij} is defined to be zero, if there exists an $r \neq i, j$, such that $\phi_r^K \neq \phi_r^{K'}$, and to be one otherwise. The constant Γ is given by

$$\Gamma = (-1)^{\sigma+\sigma'} \left(-\tfrac{1}{2}\right)^{(n-h)/2-m} (-2)^{J/2} \tag{7}$$

where n the number of electrons, h the total number of (even and odd) chains, and m is the number of cycles; J is the number of pairs for which $\phi_i^K = \phi_j^K$ but $\phi_i^{K'} \neq \phi_j^{K'}$ or vice versa; σ is the signature of the permutation of the unpaired orbitals of the ϕ_i^K back to their order in Φ_K and σ' is the signature of the $\phi_j^{K'}$ back to their order in $\Phi_{K'}$. Odd chain is abbreviated o chain, and even chain by e chain.

There are no one-electron terms from diagrams containing two even chains, and there are no terms at all from diagrams containing more than two even chains.

When there are no even chains, $q'_{ij}=1$, and if there are two even chains, $q'_{ij}=0$. When $\phi_i^K=\phi_j^K$, and/or $\phi_i^{K'}\neq\phi_j^{K'}$ $q_{ij}=0$. If there are two even chains and i and j are in the same chain, $q_{ij}=0$. Otherwise q_{ij} is given in Table 1.

3. Computational Realisation

The "best", that is the most "economic" computer algorithm has to minimize the following quantities: a) mathematical operations, b) number of processor storage locations, c) amount of data transferred to or/and from external storage, and d) number of transferred blocks of data. An algorithm that fulfills these four conditions uses the minimum of central processor and elapses time, and therefore is the cheapest. Normally, each algorithm is a compromise with these four conditions, resulting from the characteristics of the computer it is (supposed) to be implemented on.

The calculation of matrix elements between many electron wavefunctions of arbitrary spin states is especially difficult, because normally *not all* one- and two-electron integrals between the functions ϕ used to construct the wavefunction can be held in processor storage simultaneously. In this case data transfer to and from external storage becomes very critical and advanced techniques have to be applied to solve this problem. For the present problem an efficient algorithm has been designed, implemented and extensively tested. It will be described, its relation to similar algorithms will be discussed, and its present implementation will be outlined: For convenience, the number of matrix elements the numerical values of which can be formed in processor storage simultaneously is called a *core-load* of *matrix elements*, the number of symbolic matrix elements that can be held (actually: the symbolic references of which can be reordered according to one index) simultaneously on direct access external storage is called a "*disk*"-*load of matrix elements*, and the number of integrals that can be kept in processor storage (actually: directly referenced simultaneously) is called a *core-load of integrals.*

Each symbolic matrix element contains one or more references to one- and/or two-electron integrals. Each of these references is uniquely identified by two numbers: The sequence number of the matrix element it contributes to, and the sequence number of the referenced integral. To avoid time consuming searching, these references have to be ordered in such a way, that the quantities who's reference are to be resolved can be processed "sequentially". Normally, this makes one or more reorderings of the references necessary. Based on this general idea an efficient algorithm has been developed for computing numerical matrix elements which essentially consists of the following steps[2].

Step a) Compute a list of symbolic matrix elements.

Step b) Order the symbolic contributions, for a *disk-load* of matrix elements at a time, so that consecutive symbolic contributions refer to core loads of integrals in ascending sequence.

Step c) Resolve the references to the integrals.

[2] Starting from a different analysis, M. Yoshimine essentially arrived at the same result [12].

Table 2. Munich program system – Configuration interaction package release 0 (March 1973). Timing example[a]: molecular orbitals 35; configurations (all double + single sub, except for the *K*-shell) 2063; total SCF energy −76.05199 a.u.; total CI energy −76.26620 a.u.

Total storage (kBytes)	Data storage (kBytes)	Time				Δ time		Number of reads of int. list
		Step b) (min)	Step c) (min)	Step e) (min)	Total (min)	(min)	(%)	
600	453	5.7062	0.5947	0.5523	6.8573			8
540	393	5.7088	0.6025	0.5595	6.8713	0.0176	0.25	9
480	333	5.7225	0.6085	0.5762	6.9075	0.0538	0.78	11
420	273	5.7088	0.6172	0.5928	6.9192	0.0655	0.96	13
360	213	5.8029	0.6388	0.6312	7.0733	0.2196	3.2	17
300	153	5.8167	0.7287	0.7378	7.2835	0.4298	6.3	25
240	93	5.8940	0.8900	0.9220	7.7063	0.8526	12.4	43

[a] IBM 360/91.

Step d) Order the numerical contributions, so that consecutive elements refer to *core-loads* of matrix elements in ascending order. Actually, within each sequence of numerical contributions built from the *same* core-load of integrals, the elements are ordered according to matrix elements in ascending order. Therefore reordering is not necessary, if the list can be accessed directly (randomly).

Step e) Resolve the references of the numerical contributions to the matrix elements, and compute the matrix elements, a core-load at a time.

It is important to notice, that in this algorithm the list of integrals has only to be read as many times as there are *disk-loads* of matrix elements. As normally the disk (direct access) space available is rather large, one or very few reads of the integral list are necessary.

At present this algorithm has been implemented in a slightly different way: Essentially Step b) of the above sequence is applied to each *core-load* of matrix elements separately, instead to each *disk-load*. This modification of the algorithm needs relatively little disk space, approximately the order of magnitude of processor storage available for the step, and it avoids Step d) completely. But the list of integrals has to be read as many times as there are core-loads of matrix elements, which essentially means more often, because usually core-loads are smaller than disk-loads of matrix elements. But it has been found (compare Table 2) that the increase of CPU time with increasing numbers of reads of the integral list is unexpectedly small, while the elapsed time is dependent on the number of reads of the integral list, as is to be expected. Normally, the *complete* list of symbolic matrix elements is generated in Step b). This list can be used to construct the list of numerical matrix elements for any problems where the following quantities agree in number and/or type: molecular symmetry (if explicity taken into account), electrons, molecular orbitals, and configurations. But with increasing number of electrons and configurations, this list of symbolic matrix elements will become excedingly large and it might become unreasonable to keep it. In this case the above algorithm [Steps a) to e)] has to be applied to each disk-load of matrix elements separately.

In the following paragraphs some of the approaches used in the present algorithm are discussed in some more detail, to show the critical features of their performance:

A very efficient algorithm for the projective reduction of matrix elements has been described by Reeves [11], and has been implemented in the present program with minor (technical) modifications. Timing tests revealed, that the initial "pairing" of orbitals between bonded functions is very time consuming, and in the test case actually used up to 65% of the total CPU time necessary for the projective reduction. Therefore this procedure has been carefully analysed.

The procedure consists in "pairing" the orbitals between two bonded functions one-to-one so as to minimize the number of noncoincidences and to build appropriate cross-reference tables to be used in the actual projective reduction. This "pairing" may be terminated if the third noncoincidence is found, because if there are three or more noncoincident orbitals between bonded functions the matrix element between these functions is identical zero. This process of pairing has to be done for the orbitals between the first members of all orbital configurations, and in case less than three noncoincidences have been found for the orbitals, between all other members of these orbital configurations.

Two classes of approaches are possible for this "pairing", one class involving explicit searching, one class involving no searching. We have currently implemented an algorithm with searching gaining speed because it has been programmed in IBM 360 Assembler Language and is largely formed by program sections allowing the computer IBM 360/91 to run a special state (loop mode). However, we are actively investigating algorithms that do not involve searching in the hope of making further time savings [13].

Throughout the present program *linear indexing* and *table look up* has been used. In particular: all symbolic references are given as to core/disk load number and sequence number (within the load). The variables Gamma and Q are identified by entry points to appropriate tables.

Timing examples of the present program release are given in Table 2 in which the CI problem is based on a SCF problem solved for the water molecule [14].

Acknowledgement. The authors are deeply grateful to the NATO Science Committee for a grant in support of this work.

References

1. Diercksen, G. H. F.: Theoret. Chim. Acta (Berl.) **33**, 1 (1974)
2. McWeeny, R., Sutcliffe, B. T.: Methods of molecular quantum mechanics, p. 51. London: Academic Press
3. McWeeny, R.: Proc. Roy. Soc. (London) A **223**, 63 (1954)
4. Reeves, C. M.: Ph. D. Thesis, Cambridge 1957
5. Boys, S. F., Reeves, C. M., Shavitt, I.: Nature **178**, 1207 (1956)
6. Cooper, I. L., McWeeny, R.: J. Chem. Phys. **44**, 226 (1966)
7. Sutcliffe, B. T.: J. Chem. Phys. **45**, 235 (1966)
8. Roos, B.: USIP Report 72-08 (1972), Institute of Physics, University of Stockholm; Chem. Phys. Letters **15**, 153 (1972)
9. Neumann, D. B. *et al.*: Polyatom V/2 System Manual, available from QCPE. Chemistry Department, Indiana University, Bloomington, Indiana 47401, USA

10. Diercksen, G. H. F., Kraemer, W. P.: Molecular Program System, Reference Manual, Special Technical Report, Max-Planck-Institut für Physik und Astrophysik, München (To be published)
11. Reeves, C. M.: Commun. ACM **9**, 276 (1966)
12. Yoshimine, M.: J. Comp. Phys. **11**, 449 (1973)
13. de Vries, H. L., Diercksen, G. H. F.: (To be published)
14. Diercksen, G. H. F.: Theoret. Chim. Acta (Berl.) **21**, 335 (1971)

G. H. F. Diercksen
Max-Planck-Institut
für Physik und Astrophysik
D-8000 München 40
Föhringer Ring 6
Federal Republic of Germany