A NEW DETERMINANT-BASED FULL CONFIGURATION INTERACTION METHOD

P.J. KNOWLES and N.C. HANDY
University Chemical Laboratory, Lensfield Road, Cambridge CB2 1EW, UK

Received 14 August 1984; in final form 14 September 1984

Siegbahn recently presented a new direct CI method which is applicable to the CAS SCF method. Instead of using configuration state functions as adopted by Siegbahn, we propose that Slater determinants should be used. The approach then needs no formula tape, the one-electron coupling coefficients being easily generated as required. The method is totally vectorised, and applies as programmed to the lowest eigenvalue of any spin and space symmetry. The timings for Siegbahn's examples are approximately halved using this procedure, and in a full CI calculation involving 107942 configurations, 18 CRAY-1S CPU seconds were required per iteration.

1. Introduction

One of the most successful methods for the determination of high-accuracy wavefunctions is the complete active space variant (CAS SCF) of the multi-configuration self-consistent field (MC SCF) method, first realised by Roos, Siegbahn and co-workers [1]. In this method a set of orbitals are chosen as active, and all configurations which may be formed for this set, of the correct space and spin symmetry, are used in the MC SCF method. In other words a full configuration interaction (full CI) calculation is performed to obtain the CI vector at many stages of the iterative MC SCF process. The size of the full CI grows rapidly with the number of active orbitals however, and whilst useful calculations can be performed with a 1000 configuration state functions (CSFs), much more useful calculations can be performed if it is possible to use several tens of thousands of CSFs. It might then be thought that the diagonalisation time of the CI matrix would become prohibitively expensive.

Siegbahn [2] has recently addressed this problem, and has demonstrated the efficiency of a new and innovative method which can make good use of pipeline computers and thus make this part of the CAS SCF calculation no more expensive than the other parts, although there remains a considerable demand on input/output facilities. Here we show how this final problem can be eliminated through the use of determinants instead of CSFs.

If the Davidson [3] algorithm is used as the iterative procedure to obtain the required eigenvector, then the time consuming step is the evaluation of $\boldsymbol{\sigma}$, where

$$\sigma_I = H_{IJ} c_J \,, \tag{1}$$

where $\boldsymbol{c}$ is any vector, and $H_{IJ} = \langle \Phi_I | H | \Phi_J \rangle$ with Φ_I being the expansion set, usually CSFs. On introducing two-electron coupling coefficients and associated two-electron integrals, the two-electron part of eq. (1) may be written

$$\sigma_I = \sum_J \Gamma^{IJ}_{ijkl} (ij|kl) c_J \,, \tag{2}$$

where Γ^{IJ}_{ijkl} can be written in terms of unitary group one-particle generators [4] as

$$\Gamma^{IJ}_{ijkl} = \tfrac{1}{2} \langle I | E_{ij} E_{kl} - \delta_{jk} E_{il} | J \rangle \,. \tag{3}$$

The second term in (3) can be treated by absorbing combinations of two-electron integrals into the one-electron integrals, and these are processed in a standard way; the time consuming part is the evaluation of the first part. On introducing the resolution of the identity, we have

$$\Gamma^{IJ}_{ijkl} = \frac{1}{2} \sum_K \langle I | E_{ij} | K \rangle \langle K | E_{kl} | J \rangle$$
$$- \tfrac{1}{2} \langle I | E_{il} | J \rangle \delta_{jk} \,. \tag{4}$$

One-particle coupling coefficients are defined through

$$\gamma_{ij}^{IJ} = \langle I|E_{ij}|J\rangle , \tag{5}$$

and thus the time consuming part in the evaluation of σ_I may be written

$$\sigma_I = \frac{1}{2}\sum_{kl} \gamma_{kl}^{KI} \sum_{ij} (ij|kl) \sum_J \gamma_{ij}^{JK} c_J . \tag{6}$$

Siegbahn [2] recognised that this way of writing $\boldsymbol{\sigma}$ leads to a computationally efficient method for its evaluation, expressing it as

$$\boldsymbol{\sigma} = \mathrm{Tr}(\boldsymbol{\gamma}\cdot\mathbf{I}\cdot\mathbf{D}) .$$

His scheme is outlined as follows:

Loop over intermediate states $|K\rangle$

$$D_{ij}^{K} = \sum_J \gamma_{ij}^{JK} c_J , \tag{6a}$$

$$E_{ij}^{K} = \sum_{kl} (ij|kl) D_{kl}^{K} , \tag{6b}$$

$$\sigma_I = \sum_{ij} \gamma_{ij}^{IK} E_{ij}^{K} . \tag{6c}$$

Notice that the set of intermediate states $|K\rangle$ must span the full spin space (or at least the subspace which interacts through the operators E_{ij}), including CSFs of different spatial symmetry, even when the CI expansion itself is not complete. Thus the method has most to offer for complete CI calculations, and may be rather wasteful in other cases. Notice also that for a given $|K\rangle$, all γ_{ij}^{JK} must be available together. Standard CI programs, notably those based on the unitary group method, are usually driven by the orbital labels i, j, and so the formulae must be sorted and retained in external storage. Siegbahn has recognised this as the major limit of his method; in an example given in ref. [2] with 30744 CSFs, the one-electron formula tape occupied 40 megabytes of disc space, and this file has to be read in each iteration of the diagonalisation process. He recognises the desirability of a scheme which obtains the γ_{ij}^{IJ} directly in the required order, in order to reduce the elapsed time and disc overheads which on modern computers form a major part of the real cost of a calculation. Ideally, these coupling coefficients should be generated at a cost which is small relative to the central matrix multiplication in (6b), although some sacrifice of CPU efficiency is acceptable if the number of I/O operations can be reduced.

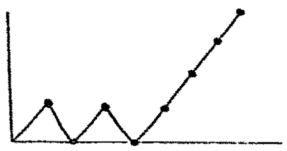
In any CI expansion consisting of CSFs, a given CSF $|K\rangle$ is related by a given excitation operator E_{ij} to a set of states $|I\rangle$, each with the same orbital occupations, but with different spin couplings. It is difficult to see how, for example, the unitary group approach could be adapted to find all these couplings for a given i, j, K in an efficient manner. For this reason we have fallen back on the idea of using an expansion set consisting of simple Slater determinants. In this case, there are at most two interacting states, and their contributions can be separated cleanly by writing the excitation operators as

$$E_{ij} = E_{ij}^{\alpha} + E_{ij}^{\beta} , \tag{7}$$

where E_{ij}^{α} excites one electron from α spin orbital j to α spin orbital i.

In deciding to use determinants, rather than spin symmetry adapted functions, we have to be aware of the possible disadvantages. The most obvious difficulty is that there are typically two or three times as many Slater determinants as CSFs in a CI expansion of given spin symmetry, and so we have to be able to store a much large CI vector; in addition the number of intermediate states $|K\rangle$ in (6) is increased. We recognise this as the price which has to be paid in adopting our approach.

It may happen that the lowest state of the required spin (S_z and S^2) symmetry is not the lowest in a basis of determinants. Thus simply finding the lowest eigenvector of the Hamiltonian matrix in the determinant basis is not a satisfactory procedure for finding the ground state wavefunction of a given spin symmetry. In our programs, this problem is dealt with in the following way. The initial guess for the eigenvector is taken as the single determinant of lowest energy with the S_z quantum number equal to the required S. If this consists of doubly occupied orbitals and just $2S_z$ alpha spin occupied orbitals, then it is an eigenfunction of S^2 with the correct eigenvalue, and no further action need be taken. Otherwise, it is a mixture of several functions with different S^2 eigenvalues. We then construct a spin eigenfunction of the correct spin and involving this determinant; computationally, the simplest is the $X(N, S, S, 1212...111...)$ branching diagram function [5], e.g.



Having obtained a trial vector c_0 of correct symmetry, this symmetry can be maintained throughout the Davidson iterative process. Because there are no spin operators in the Hamiltonian, σ will have the same symmetry as c_0. There will be no numerical round off problem because exactly the same integrals contribute from each determinantal component of the spin eigenfunction. In the Davidson process, a new trial vector is calculated by

$$\Delta c_I = \sigma_I/(E - H_{II}) . \tag{8}$$

To maintain spin symmetry, H_{II} has to be replaced by an average diagonal energy of all those determinants with the same orbitals being singly occupied. This should cause no change in the efficiency of the Davidson process.

We have used a very simple procedure to calculate these average diagonal energies, which appears to work well in the examples we have tried. A formula for the exact diagonal energies is

$$\frac{1}{2}\sum n_i^\alpha n_j^\beta J_{ij} + \frac{1}{2}\sum n_i^\alpha n_j^\alpha J_{ij} + \frac{1}{2}\sum n_i^\beta n_j^\beta J_{ij}$$

$$+ \sum (n_i^\alpha + n_i^\beta)\bar{h}_{ii} + \frac{1}{2}\sum K_{ij} n_i^\alpha (1 - n_j^\alpha)$$

$$+ \frac{1}{2}\sum K_{ij} n_i^\beta (1 - n_j^\beta) , \tag{9}$$

where n_i^α, n_i^β are occupation numbers and

$$\bar{h}_{ii} = h_{ii} - \frac{1}{2}\sum_k (ik|ik) . \tag{10}$$

To obtain approximate average diagonal energies which have the above requirement, we replace all the exchange integrals K_{ij} occurring in the last two terms of (9) by the maximum exchange integral occurring in the integral list; this ensures that all the diagonal energies are upper bounds to the true values.

There are obviously other ways of achieving the requirement, but they may take a longer time to evaluate.

We experienced no problems of spin contamination in a calculation using this approach on the $^3\Sigma^+$ and $^1\Sigma^+$ states of MnH. With the basis set and geometry which we used, the $^5\Sigma^+$ state is the lowest eigenvector in a basis of determinants with $S_z = 0$, 1 and 2, yet we were able to obtain all the required $S_z = S$ eigenvectors as the lowest states in the Davidson procedure.

2. The determinant CI algorithm

The advantage of using determinants as the expansion functions in full CI calculations was first shown by Handy [6]; the idea has enabled many benchmark full CI calculations to be performed [7]. In those applications the Cooper–Nesbet [8] diagonalisation algorithm was used, because that needs the storage of only one CI vector, but here the Davidson cheme must be used, and the problem in mind is the iterative CAS SCF method. In all of what follows, non-degenerate point group symmetry can be exploited to the full. Its treatment is straightforward, and we omit details for the sake of clarity.

Any Slater determinant $|K\rangle$ can be written as the product of a "string" Φ^α of occupied alpha orbitals and a "string" Φ^β of occupied beta orbitals. Since the configuration expansion consists of all determinants with the same S_z quantum number, then the sets of strings consist of all those that can be formed from the same number of electrons.

The addressing of the CI vector is achieved in the following way. We define, separately for alpha and beta strings, an addressing array Z, given by

$$Z(k, l) = \sum_{m=M-l+1}^{M-k} \left[\binom{m}{N-k} - \binom{m-1}{N-k-1}\right]$$

$$(M - N + k \geqslant l \geqslant k \, ; k < N) ,$$

$$Z(N, l) = l - N \quad (M \geqslant l \geqslant N) , \tag{11}$$

where k refers to an electron, l to an orbital, M is the number of orbitals, and N the number of electrons. Any string is identified by a list of occupied orbitals $\{\phi_i; i = 1, 2, ... N\}$ in strictly ascending order, i.e. $\phi_1 > \phi_2 > \phi_3$ ... The address of the string is then given by

$$\Phi = 1 + \sum_{i=1}^{N} Z(i, \varphi_i) \qquad (12)$$

and with the above definition of Z, the addressing is lexical without gaps. For example, if $M = 5$ and $N = 3$, the ten possible strings occur in the order (123), (124), (125), (134), (135), (145), (234), (235), (245), (345).

Given the addresses of the alpha and beta strings Φ^{α}, Φ^{β} forming a given Slater determinant, the determinant can be addressed simply as the element of a rectangular array $C(\Phi^{\beta}, \Phi^{\alpha})$. In this way, operations on the alpha string alone can be performed for all beta strings in a vector loop, and vice versa. Hence this addressing scheme, which is however applicable only to a complete CI, is ideally suited to pipeline computers.

The one-particle matrix elements γ_{ij}^{IK} involving the intermediate states must thus be

$$\langle\Phi^{\alpha}\Phi'^{\beta}|E_{ij}^{\beta}|\Phi^{\alpha}\Phi^{\beta}\rangle \quad \text{and} \quad \langle\Phi'^{\alpha}\Phi^{\beta}|E_{ij}^{\alpha}|\Phi^{\alpha}\Phi^{\beta}\rangle, \qquad (13)$$

where Φ'^{β} is a beta string related by a single excitation to Φ^{β}. Φ'^{α} is similarly related to Φ^{α}. We may generate all the matrix elements by constructing two lists (one for α, one for β) for each K (i.e. $\Phi^{\alpha}\Phi^{\beta}$) of all single replacements. Each entry in the list contains three integers: the lexical address of Φ', the numerical value of the matrix element which is +1 or −1, and ij. Although construction of these lists is unvectorisable, each list is used many times. It may be necessary to split the lists if it is not possible to hold the full lists and the associated D^K and E^K matrices in the available memory. This is achieved by splitting the strings into blocks, and treating simultaneously only those strings which lie in the current block.

The scheme for the evaluation of (6a), (6b), (6c) on a vector computer such as the CRAY is outlined in appendix 1. It will be seen that the procedure is wholly vectorised in its time consuming parts, and that there is no I/O required. The storage requirements are for two CI vectors, the lists of single replacements, and the matrices **D**, **E** for a reasonable number of α and β strings (63 is optimal) in each block. In the largest example quoted below, 400000 words of memory were used, which is not excessive. A little more store would have increased the efficiency of the processing of the γ_{ij}^{IK} (but not the matrix multiplication) by perhaps 10%. In the CRAY-1S available to us with 1 Mword of memory, it will be possible to consider CAS SCF calculations with 300000 determinants (or about 100000 spin-adapted functions). The program which implements appendix 1 is extremely simple, and is in total less than 1000 lines of standard FORTRAN together with just one assembler matrix multiplication routine for stage (e). As indicated in ref. [2], it is important that this matrix multiplication routine evaluates and compares the sparsities and vector lengths for the two possible ways of performing the matrix multiplication. In particular, if starting with a sparse vector c, **D** is very sparse; later in the calculation, this is not so, and one should take advantage of the greater vector length obtained by varying $\Phi^{\alpha}\Phi^{\beta}$ rather than ij in the inner loop.

We note at this stage that for singlet wavefunctions, further savings in effort are possible. As pointed out by Handy [6], one may use, instead of the set of all determinants $\Phi^{\alpha}\Phi^{\beta}$, the expansion set $\{\Phi_i^{\alpha}\Phi_i^{\beta};\ 2^{-1/2}(\Phi_i^{\alpha}\Phi_j^{\beta} + \Phi_j^{\alpha}\Phi_i^{\beta})\}$. In this way, the number of intermediate states could be reduced, with some degradation of the efficiency of stages (c), (d), (f), (g). In the examples which we quote, this symmetry has not been exploited, and single determinants have been used throughout.

In table 1, details are given of the application of this procedure to a case of the same size as the largest considered by Siegbahn, a full CI involving 30744 CSFs (in our case 106820 determinants). It is seen that the total time per iteration was less than 10 s. In a typical CAS SCF calculation, it may be necessary to perform 20 applications of this procedure, indicating that one can expect to perform CAS SCF calculations of this size within 5 min of CPU time (assuming that integral transformation and other operations do not account for a large fraction of the cost).

When applying this method on a scalar machine, it is probably best to use a slightly different algorithm which uses less store, and forms a list only of beta single replacement information. The details are in appendix 2. In table 2, comparisons are made between the use of the two approaches for a smaller case consisting of 9800 CSFs (again identical in size to one of Siegbahn's examples). On the CRAY, we see that the method of appendix 1 is preferable. On the scalar IBM 3081D (scalar speed about half that of the CRAY), the matrix multiplication is the rate determining step,

Table 1
CRAY-1S CPU times (in s) per iteration for a 30744 CSF full CI calculation on O_3 [a)]

Total CPU	a	b	c	d	e [b)]	f	g
4.7–8.4	0.16	0.90	0.81	0.61	1.1–4.7	0.56	0.59

No. of determinants: 106820; No. of intermediate determinants $|K\rangle$: 213444

a) a, construction of lists of single replacements Φ'^{α}; b, construction of lists of single replacements Φ'^{β}; c, construction of $\mathbf{D}$ arising from E^{α}_{ij}; d, construction of $\mathbf{D}$ arising from E^{β}_{ij}; e, matrix multiplication $\mathbf{E} = \mathbf{ID}$; f, construction of $\boldsymbol{\sigma}$ from E^{α}_{ij}; g, construction of $\boldsymbol{\sigma}$ from E^{β}_{ij}.
b) In e, the earlier iterations are faster due to the sparsity of the CI vector. Active orbitals: $4a_1-11a_1$, $3b_2-5b_2$; 10 electrons

and the slight loss of efficiency of the method of appendix 2, therefore, insignificant. We have also performed this calculation using our original determinant program, as well as with our present MC SCF program [9], which uses CSFs and constructs a formula tape. For MC SCF calculations, where a large number of CI iterations have to be performed, construction of a formula tape has been the most efficient approach up to date. We see that in this example our new scheme is faster by a factor of 7 in CPU time, without counting the formula generation time, and that since the formula tape is of length 60 Mbyte, this calculation virtually represents the limit in size for the conventional approach.

As a final example we have performed a full CI calculation on the $X\,^2A_1$ state of NO_2, including all the valence orbitals plus an additional orbital of a_1 symmetry. This yielded a CI expansion of 230470 determinants, equivalent to 107942 CSFs, and the calculation required 18 CPU seconds per iteration on the CRAY. This calculation probably represents the limit in size of what may be done with our program at present (800000 words of storage available). Larger calculations are possible, since the lexical ordering of the CI vector means that in the processing of blocks of alpha and beta strings in appendix 1, one need only hold in the memory every determinant whose alpha string lies in the current alpha block, and every determinant whose beta string lies in the current beta block. In this way, less memory is required, at the expense of performing I/O operations. We have not pursued this option, since it may be rather difficult to incorporate it into our MCSCF program.

Table 2
CPU times per iteration for a 9800 CSF full CI calculation on O_3

Computer/method	Total CPU (s)	Breakdown (as table 1)						
		a	b	c	d	e	f	g
CRAY/App. 1	1.0–1.8	0.08	0.16	0.16	0.10	0.4–1.2	0.11	0.09
CRAY/App. 2	5.0–5.8	0.07	0.08	0.13	2.15	0.4–1.2	0.06	2.12
IBM3081D/App. 1	14–48	0.1	0.8	1.7	2.5	6–40	0.9	2.0
IBM/App. 2	15–49	0.2	0.1	1.6	3.1	6–40	0.8	3.0
CRAY/formula tape [9]	13.5	(5 min to generate 60 Mbyte formula tape)						
IBM/determinants [6]	86							
IBM/singlet determinants [6]	45 [a)]							

No. of determinants: 31752; No. of intermediate determinants $|K\rangle$: 63504

a) Using the functions $2^{-1/2}(\Phi^{\alpha}_{i}\Phi^{\beta}_{j} + \Phi^{\alpha}_{j}\Phi^{\beta}_{i})$.

3. Conclusion

The previous section shows that the combination of Siegbahn's idea and the use of determinants instead of SCFs leads to an extremely efficient full CI code.

Adopting determinants has meant an increase in the number of intermediate states, and therefore an increase in the cost of the matrix multiplication (6b); however, the treatment of the coupling coefficients is considerably simplified and may be performed without any use of disc storage. This is extremely important when using modern hardware, where CPU processing is much cheaper than disc access. This full CI code is at present being incorporated into our current MC SCF program. In the context of direct quadratically convergent MC SCF [10], as well as constructing $\mathbf{H}c$ our current MC SCF program requires the construction of a transition density matrix of the form $c_0^\dagger \Gamma c_1$ simultaneously with a product of the form $\mathbf{H}_0 c_1 + \mathbf{H}_1 c_0$. It is straightforward to adapt our scheme to do this efficiently. We then expect to report CAS SCF calculations with more than 100000 CSFs in the near future.

Acknowledgement

We are grateful to Professor Siegbahn for a preprint of his paper. This work relies completely on that paper, and we recognise its original contribution. We wish to thank Mr. R.J. Harrison for helpful discussions.

Appendix 1. The determinant full CI algorithm, for a vector machine, for the evaluation of $\boldsymbol{\sigma}$, eq. (6)

```
Split $\alpha$ strings and $\beta$ strings into blocks

Loop over blocks of $\alpha$ strings
(a) Loop over $\alpha$ strings $\Phi^\alpha$ in block
    Form list of single replacements $\Phi'^\alpha = E_{ij}\Phi^\alpha$

Loop over blocks of $\beta$ strings
(b) Loop over $\beta$ strings in block
    Form list of single replacements $\Phi'^\beta = E_{ij}\Phi^\beta$

(c) Loop over $\Phi^\alpha$ in block
    Loop over $\Phi'^\alpha$
    Loop over $\Phi^\beta$ in block; $|K\rangle = |\Phi^\alpha\Phi^\beta\rangle$, $|J\rangle = |\Phi'^\alpha\Phi^\beta\rangle$
      $D(\Phi^\alpha, \Phi^\beta, ij) = D(\Phi^\alpha, \Phi^\beta, ij) \pm C(\Phi'^\alpha, \Phi^\beta)$;
vectorised

(d) Loop over $\Phi^\beta$ in block
    Loop over $\Phi'^\beta$
    Loop over $\Phi^\alpha$; $|K\rangle = |\Phi^\alpha\Phi^\beta\rangle$, $|J\rangle = |\Phi^\alpha\Phi'^\beta\rangle$
      $D(\Phi^\alpha, \Phi^\beta, ij) = D(\Phi^\alpha, \Phi^\beta, ij) \pm C(\Phi^\alpha, \Phi'^\beta)$;
vectorised

(e) $E(\Phi^\alpha, \Phi^\beta, ij) = \Sigma_{kl}(ij|kl) * D(\Phi^\alpha, \Phi^\beta, kl)$; matrix
multiply vectorised

(f) Loop over $\Phi^\alpha$ in block
    Loop over $\Phi'^\alpha$
    Loop over $\Phi^\beta$ in block; $|K\rangle = |\Phi^\alpha\Phi^\beta\rangle$, $|I\rangle = |\Phi'^\alpha\Phi^\beta\rangle$
      $\sigma(\Phi'^\alpha, \Phi^\beta) = \sigma(\Phi'^\alpha, \Phi^\beta) \pm E(\Phi^\alpha, \Phi^\beta, ij)$;
vectorised

(g) Loop over $\Phi^\beta$ in block
    Loop over $\Phi'^\beta$
    Loop over $\Phi^\alpha$; $|K\rangle = |\Phi^\alpha\Phi^\beta\rangle$, $|I\rangle = |\Phi^\alpha\Phi'^\beta\rangle$
      $\sigma(\Phi^\alpha, \Phi'^\beta) = \sigma(\Phi^\alpha, \Phi'^\beta) \pm E(\Phi^\alpha, \Phi^\beta, ij)$; vectorised
```

Appendix 2. The determinant full CI algorithm, for a scalar machine, for the evaluation of $\boldsymbol{\sigma}$, eq. (6)

```
Split $\beta$ strings into blocks

Loop over blocks of beta strings
(b) Loop over beta strings $\Phi^\beta$ in block
    Form list of single replacements $\Phi'^\beta = E_{ij}\Phi^\beta$

Loop over all alpha strings
(a) Form single replacements $\Phi'^\alpha$

(c) Loop over $\Phi'^\alpha$
    Loop over $\Phi^\beta$ in block
      $D(\Phi^\alpha, \Phi^\beta, ij) = D(\Phi^\alpha, \Phi^\beta, ij) \pm C(\Phi'^\alpha, \Phi^\beta)$

(d) Loop over $\Phi^\beta$ in block
    Loop over $\Phi'^\beta = E_{ij}\Phi^\beta$
      $D(\Phi^\alpha, \Phi^\beta, ij) = D(\Phi^\alpha, \Phi^\beta, ij) \pm C(\Phi^\alpha, \Phi'^\beta)$
```

```
(e) Multiply $E(\Phi^\alpha, \Phi^\beta, ij) = \Sigma_{kl}(ij|kl)*D(\Phi^\alpha, \Phi^\beta, kl)$

(f) Loop over $\Phi'^\alpha$
    Loop over $\Phi^\beta$ in block
    $\sigma(\Phi'^\alpha, \Phi^\beta) = \sigma(\Phi'^\alpha, \Phi^\beta) \pm E(\Phi^\alpha, \Phi^\beta, ij)$

(g) Loop over $\Phi^\beta$ in block
    Loop over $\Phi'^\beta = E_{ij}\Phi^\beta$
    $\sigma(\Phi^\alpha, \Phi'^\beta) = \sigma(\Phi^\alpha, \Phi'^\beta) \pm E(\Phi^\alpha, \Phi^\beta, ij)$
```

References

[1] P.E.M. Siegbahn, J. Almlöf, A. Heiberg and B.O. Roos, J. Chem. Phys. 74 (1981) 2384;
B.O. Roos, P.R. Taylor and P.E.M. Siegbahn, Chem. Phys. 48 (1980) 157.
[2] P.E.M. Siegbahn, Chem. Phys. Letters 109 (1984) 417.
[3] E.R. Davidson, J. Comput. Phys. 17 (1975) 84.
[4] J. Paldus, J. Chem. Phys. 61 (1974) 5321.
[5] R. Pauncz, Spin eigenfunctions (Plenum, New York, 1979).
[6] N.C. Handy, Chem. Phys. Letters 74 (1980) 280.
[7] R.J. Harrison and N.C. Handy, Chem. Phys. Letters 95 (1983) 386;
P. Saxe, H.F. Schaefer III and N.C. Handy, Chem. Phys. Letters 79 (1981) 202.
[8] R.K. Nesbet, J. Chem. Phys. 43 (1965) 311.
[9] P.J. Knowles, G.J. Sexton and N.C. Handy, Chem. Phys. 72 (1982) 337.
[10] B.H. Lengsfield III, J. Chem. 77 (1982) 4073.