Title
THE SHAPE-DRIVEN GRAPHICAL UNITARY GROUP APPROACH TO LARGE-SCALE CONFIGURATION INTERACTION CALCULATIONS

Author
Saxe, P.W.

Publication Date
1982-09-01

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials & Molecular Research Division

THE SHAPE-DRIVEN GRAPHICAL UNITARY GROUP APPROACH TO LARGE-SCALE CONFIGURATION INTERACTION CALCULATIONS

Paul William Saxe
(Ph. D. thesis)

September 1982

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

THE SHAPE-DRIVEN GRAPHICAL UNITARY GROUP APPROACH TO LARGE-SCALE CONFIGURATION INTERACTION CALCULATIONS

By

Paul William Saxe
(Ph.D. Thesis)

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

September 1982

This research was partially supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Chemical Sciences Division of the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

The Shape-Driven Graphical Unitary Group Approach to Large-Scale Configuration Interaction Calculations

By

Paul William Saxe

Lawrence Berkeley Laboratory, University of California
Berkeley, California 94720

Abstract

A new algorithm has been developed for large-scale configuration interaction (CI) calculations. This new approach, called the Shape-Driven Graphical Unitary Group Approach (SDGUGA), was designed to overcome some of the defects inherent in the previous Loop-Driven Approach (LDGUGA). The Shape-Driven Approach leads to a direct CI program based on the simplification of the external space for calculations involving only single and double excitations from a multi-reference configuration set. By exploiting the shapes of the external portions of loops on the Shavitt graph as well as the tremendous structure implicit in the unitary group approach, the construction of Hamiltonian matrix elements is reduced to an insignificant portion of a calculation. Therefore, programs based on the SDGUGA are considerably more efficient than previous programs. This efficiency is the key to the success of the direct CI

formulation, and also allows the computation to be restructured so that the entire CI and correction vectors need not be held in the central memory of the computer. The Shape-Driven Approach is the first CI method capable of extremely large, general CI calculations. This ability is demonstrated by various calculations, including one calculation with a 703 configuration reference state and over one million configurations in the CI expansion. The present algorithm is also shown to be well suited to a vector computer such as the CRAY.

Table of Contents

Acknowledgements

This work would not have been possible without the constant support, encouragement and enthusiasm of Fritz Schaefer. Fritz always had confidence that we could indeed run a calculation with a million configurations, even at the beginning when the idea seemed a bit far-fetched. I am also most grateful to Nick Handy who was the first to truly grasp the implications of my suggestion of a shape-driven approach. Nick also did an amazing job of programming the external shapes for the original program, generating almost flawless code at a prodigous rate.

The success of the SDGUGA program is based on its ability to page the vectors. For this, I am greatly indebted to Doug Fox, who managed to master the intricate detail needed to convert the proposed algorithm into a functioning program. The resulting program is a tribute to Doug's tenacity. Bernie Brooks first introduced me to the Unitary Group Approach, and wrote the LDGUGA programs on which the SDGUGA is based. Bernie along with Bill Laidig also taught me a great respect for efficient programs. I enjoyed many lengthy discussions with Bill, and it was during these that the first seeds of the shape-driven algorithm were sown.

I would also like to thank Steve Binkley for all of his work in getting the program running on the CRAY -- and for being interested in vectorizing a CI program in the first place.

Finally, I am grateful for the wonderful atmosphere of Fritz's group of students and of the chemistry department at Berkeley as a whole. My wife, Susan, gets the last thanks for her support and for her work on this thesis.

For the last year of my studies, my support was provided by a Merck Predoctoral Fellowship. This allowed me much of the freedom necessary for the completion of the SDGUGA CI program, for which I am very grateful.

This research was partially supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Chemical Sciences Division of the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

I. INTRODUCTION

Ab initio quantum chemistry is the attempt to investigate chemical problems without recourse to experimental data, apart from fundamental physical constants such as the speed of light. In principle, by solving the Schroedinger equation or the relativistic counterparts such as the Dirac equation, one can obtain an exact knowledge of the potential energy hypersurface of a chemical system, from which it would appear that almost all of chemistry can be understood. While it is difficult to know absolutely if such equations accurately model nature, there is a large body of evidence to suggest that for all practical purposes, they are sufficient.

The Schroedinger equation, however, can only be analytically solved for a few systems. For molecules and atoms, only the one-electron atoms lead to soluble equations. For a molecule such as water or methane, the exact solutions cannot be found, so the task of the quantum chemist is to find approximate solutions of sufficient accuracy to answer chemical problems. Several approximations are commonly made in such calculations. Each will be commented on in turn, and they will be arbitrarily grouped into two classes: those in which the physical model is modified and those involving truncations of mathematical expansions.

The first class of approximations are based on physical intuition and will result in solutions which must differ from the true solution of the problem. The first approximation of this

sort which is made is the use of the nonrelativistic Schroedinger equation. The bulk of quantum chemical calculations to date involve such relatively light elements as carbon and hydrogen for which the relativistic effects are expected to be relatively insignificant[1]. For this reason, the starting point of most calculations has been the Schroedinger equation, often with no comment about the tacit assumption. In heavier atoms, where the relativistic effects are no longer small, they can either be treated as a perturbation, using a Pauli-Breit Hamiltonian,[2] for instance; or the Dirac equation can be used, leading to Dirac-Hartree-Fock[3] procedures.

A second major approximation made is the Born-Oppenheimer or fixed-nucleus approximation.[4] For a molecule of more than one atom, it is a formidable problem to solve for the simultaneous motion of the electrons and nuclei. Since the electrons are much less massive than the nuclei and therefore move much faster, it is reasonable to assume that the electrons instantaneously rearrange upon a perturbation of the nuclear framework. As a result, the nuclear and electronic wavefunctions are separable. The electronic Schroedinger equation is solved for a number of nuclear geometries to give a potential surface, which can then be used as the potential term to solve for the motion of the nuclei.

Empirically, both of these approximations are very effective, reducing the complexity of the problem greatly without adversely affecting the quality of the results. The relativistic effects must be incorporated when dealing with heavier atoms such as

silver or lead. It is possible that they are significant for elements as light as nickel[5], but that depends greatly on the accuracy desired. The Born-Oppenheimer approximation is excellent in all normal molecules, but would be suspect in calculations on exotic molecules with electrons replaced by more massive particles such as muons.

The final approximation usually made in the model concerns the treatment of the electrostatic repulsion between the electrons. The instantaneous repulsion is difficult to treat, so the Hartree-Fock approximation[6] treats the motion of each electron in the average field of the other electrons. Techniques exist for the treatment of this correlation between the motions of the electrons, and they will be commented on presently. The results of Hartree-Fock calculations can be not only quantitatively but also qualitatively in error.[7] The approximation is generally least severe for normal molecules near their equilibrium geometries. Even then, caution must be used. For example, the splitting between the lowest singlet and triplet states of methylene is overestimated by about 15 kcal[8] at the Hartree-Fock level while the F_2 molecule is found to have a negative dissociation energy.[9] The approximation usually fares less well in situations where chemical bonds are stretched or being broken or formed. Thus reaction barriers, for example, are likely to be in error in the Hartree-Fock picture.

The second class of approximation involves the truncation

of expansions, which means that as accurate an answer as desired can in principle be obtained. The Hartree-Fock equations are in general insoluble analytically but approximate solutions can be obtained by expanding the wavefunction in a set of basis functions. The Hartree-Fock equations can then be solved using the iterative self-consistent field (SCF) method. By using a complete set of basis functions, which in this case would be an infinite set, the exact solution could be obtained. In practice, it is possible, particularly for smaller systems, to approach quite closely this basis set limit.

To go beyond the level of a Hartree-Fock calculation and attempt to recover the correlation energy, one must resort to something like a configuration interaction[10] (CI) or perturbation theory calculation. The correlation energy arises from the instantaneous correlation of the motion of the various electrons and is defined as the difference between the exact nonrelativistic result and the Hartree-Fock answer. The concept of a CI calculation is to solve the Schroedinger equation approximately by expanding the wavefunction in a set of orthonormal n-electron functions called configurations. The configurations are formed by exciting electrons from occupied orbitals of the Hartree-Fock configuration into virtual orbitals. The full set of configurations corresponding to all possible arrangements of the electrons on the orbitals defined by the SCF procedure tends to be an enormously large number. Therefore, it is common practice to truncate the configuration

list after all of the single and double excitations from the Hartree-Fock reference have been included.

A calculation with such a truncated list is referred to as CI with singles and doubles (CISD) and represents a reasonable compromise between the size of the calculation and the accuracy of the results. Since the Hamiltonian contains only two-body interactions, configurations which are higher than a double excitation from the Hartree-Fock reference cannot directly interact with the reference and are therefore less important. In some senses, this truncation of the CI configuration list is comparable to the truncation of the perturbation expansion. Even when restricted to singles and doubles, the configuration list grows as the fourth power of the size of the system treated, and the computational effort involved in the CI calculation, as the sixth power.

The CI problem is similar to the SCF procedure in that if the CI expansion is not truncated -- a calculation called a full-CI -- the result is the exact solution of the electronic Schroedinger equation within the basis set used for the SCF calculation. For an infinite basis, a full-CI results in the exact solution of the Schroedinger equation.

Both Hartree-Fock and CI calculations are variational, so the energies obtained are upper limits to the true energies. Since the energies must converge to the true answers as the size of the basis set increases and the length of the CI expansion also increases, there is a certain element of truth to the statement

"the bigger, the better". A larger basis or more extensive CI will result in lower, hence better, energies.

The current work is concerned with the computational aspects of large-scale CI calculations. At the time of this writing, calculations with more than a few tens of thousands of configurations are not common. The methods described in this work are capable of efficiently handling more than a million configurations on a minicomputer, which would suggest that ten or more million configurations should be manageable on a mainframe.

Some objections may be raised as to the necessity of such large configuration lists; objections which in some ways are valid. However, for larger molecules, millions of configurations may be necessary just to do a CISD calculation from the Hartree-Fock reference. As an example, a CISD calculation on the norbornyl cation ($C_7H_{11}^+$) with no molecular symmetry and a standard 6-31G basis has about 600,000 configurations even if the carbon 1s core orbitals are not correlated and the corresponding virtual orbitals deleted from the calculation. The addition of polarization functions on only the carbon atoms to give a 6-31G* basis boosts the number of configurations to 1.6 million.

Presently, there is no method other than a direct CI[11] which can handle such a large calculation and still give reasonable results. One could try to select important configurations, but any method to handle an arbitrary configuration list will spend a

considerable amount of time forming the Hamiltonian matrix, which means that it is desirable to store the matrix. Yet for any reasonable number of configurations, the matrix is simply too large to store. Therefore, a configuration selection technique would either have to so severely restrict the number of configurations that the answers would be questionable, or would have to regenerate the Hamiltonian matrix as in a direct CI. The latter approach would probably take more computer time than doing the entire unselected calculation!

Another reason to use large configuration lists is to try to approach the full-CI limit by including higher excitations either directly[12] or by using many references.[8b] Such calculations can be needed to try to reach an accuracy of about 1 kcal in probing a chemical problem. They are also of theoretical interest as a check of the validity of the approximations made in other calculations.

Finally, the ability to do large-scale calculations will, hopefully, lead to a better understanding of the characteristics peculiar to larger calculations. Hopefully, these characteristics can be exploited to avoid a least part of the computational effort now required. One possibility worth investigating is that for extended systems many of the Hamiltonian matrix elements may be of negligible magnitude since correlation effects are presumably rather short range effects. One would not expect to encounter such accidental zeroes in a calculation on water or methane, but

perhaps they could be important in a system such as biphenyl.

In the meantime, the quantum chemist interested in understanding chemical applications should benefit greatly from the ability to undertake more accurate calculations on smaller systems or correlated calculations of any sort on larger systems. Such calculations will require large amounts of computer time, but they are now feasible.

II. REVIEW OF UNITARY GROUP APPROACH

In practice, the solution of the CI problem can be divided into two parts: the generation of a list of configurations in terms of which the wavefunction will be expanded, and the solution of the eigenvalue problem. Traditionally, the configurations have been expressed as a linear combination of Slater determinants, in which case the matrix elements H_{ij} needed for the solution of the eigenvalue problem are given by quite simple formulae. The unitary group approach[13] (UGA) directly provides a convenient basis for the expansion of the wavefunction. The Gelfand states are an orthonormal spin-adapted set of n-electron functions -- that is, they can be directly used as configurations. Furthermore, the work of Paldus, Shavitt, Drake, Schlesinger[14], and others has led to a simple and efficient scheme for the evaluation of the necessary matrix elements.

Since the advances described in the current work are based on the unitary group approach, and in particular on the structure inherent in the approach, the following sections will attempt to provide an introduction to the formalism of the UGA. This introduction is necessarily brief and will highlight those aspects which will be useful in understanding later developments; therefore, the reader is directed especially to the review articles of Paldus[14a] and Shavitt[14b] for further details as well as other references. A great deal of emphasis will be placed on the graphical aspects of the GUGA since it is through these aspects that the structure inherent in the in the UGA is most apparent; however, the reader should realize that

the graphical UGA is not different from the UGA. Rather, it is a heuristic tool intended to aid in the understanding of the structure within the UGA.

A. Configurations

The Gelfand states, as has been mentioned, are orthonormal spin-adapted n-electron functions, formed by sequentially coupling n orbitals to give a total spin quantum number S. Each function can be represented as a Paldus tableau:

$$[P] = \begin{bmatrix} a & b & c \\ n & n & n \\ a & b & c \\ n-1 & n-1 & n-1 \\ . & . & . \\ . & . & . \\ . & . & . \\ a & b & c \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad (1)$$

The values in the k^{th} row give the number of electrons, N_k, and spin, S_k, after coupling the first k orbitals, as follows:

$$\begin{aligned} N_k &= 2a_k + b_k \\ S_k &= b_k / 2 \end{aligned} \qquad (2)$$

The third column is in some senses redundant if k is known, since

$$a_k + b_k + c_k = k \qquad (3)$$

and will not be referred to hereafter.

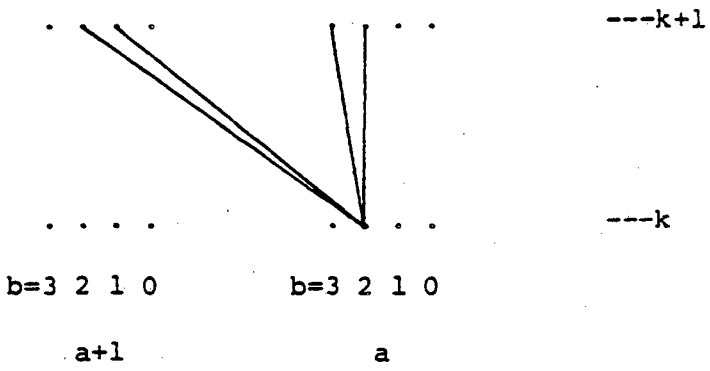
As a result of the Pauli principle, there are only four different valid rows at the level k+1 given a particular row k. These four rows correspond to the $(k+1)^{th}$ orbital being unoccupied; singly occupied and coupled to increase the total spin; singly occupied and coupled to decrease the spin; and finally doubly occupied. These four ways of getting from one level to the next are presented in the following table and given a case value, s.

Table 1. Case Values

s_k	a_k	b_k	c_k	N_k	S_k
0	0	0	1	0	0
1	0	1	0	1	+1/2
2	1	-1	1	1	-1/2
3	1	0	0	2	0

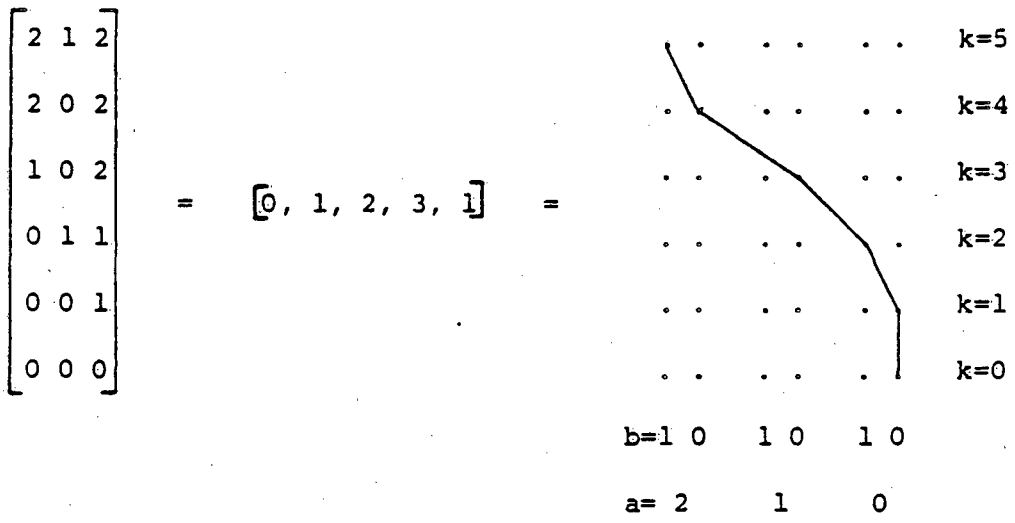
In the graphical representation, the four case values are denoted by varying inclinations of line segments as in the following diagram. Note that both a and b values are given by the lateral position of the vertices.

Figure 1. Graphical representation of four case values.



Given these definitions, a configuration can be represented as a Paldus tableau, as a vector of case values [s], or as a walk on a graph. The following example will illustrate this:

Figure 2. Equivalence of Paldus tableau, step vector and graph.

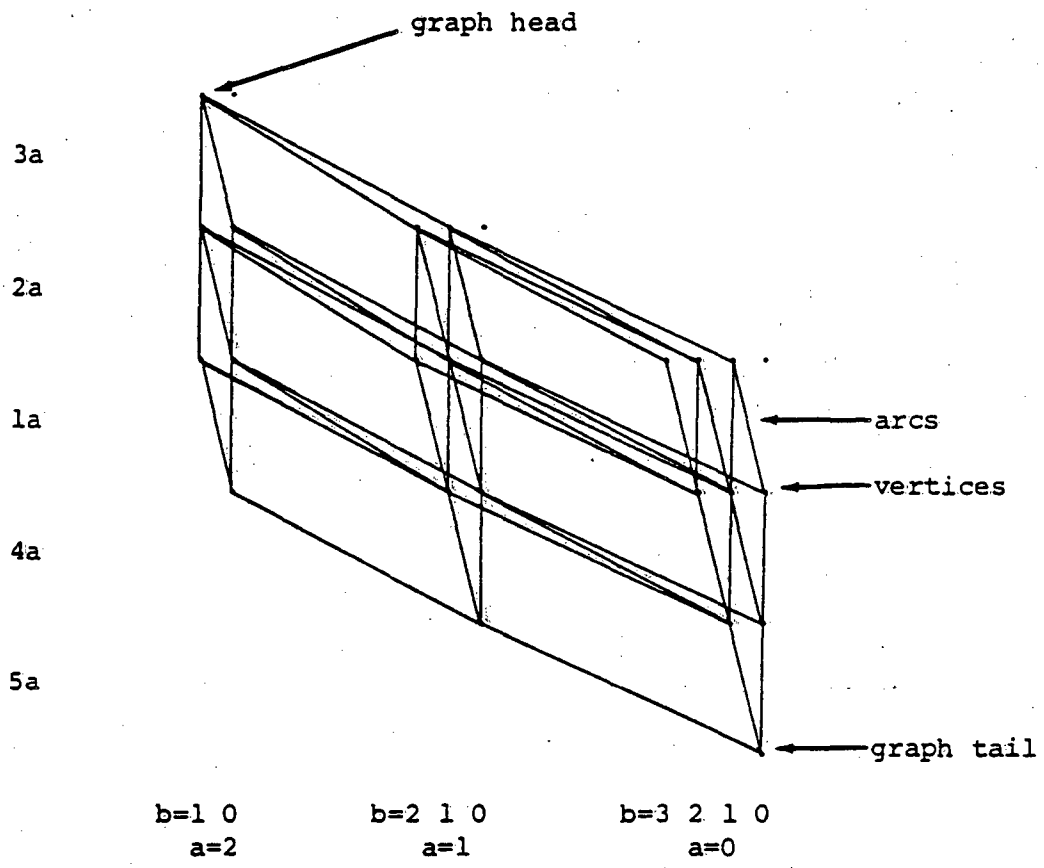


In the graph of Figure 2, the Gelfand state is represented as a series of arcs joining vertices to form what is usually referred to as a walk or configuration, even though it only <u>represents</u> a configuration. All

three representations in Figure 2 correspond to the same configuration -- one with five electrons and five orbitals, coupled to form a doublet state. Orbitals 2, 3, and 5 are singly occupied; 1 is unoccupied; and 4, doubly occupied.

If we represent all Gelfand states for 5 electrons in 5 orbitals coupled to a doublet graphically, and superimpose them on the same graph, the result is Figure 3:

Figure 3. Sample Shavitt graph for full CI for 5 electrons in 5 orbitals, coupled to a doublet.



At first glance it may be somewhat surprising that all the walks coincide at both the graph head and tail. However, the bottom row of all Paldus tableaus is always (0 0 0) and hence all the walks must coincide at the graph tail. Furthermore, the top row of the tableau is completely determined by the number of orbitals and electrons as well as the total spin. For the example in Figure 3, the top row is (2 1 2) since from Eqtn 2,

$$\begin{aligned} b_n &= 2\ S \\ a_n &= N\ /\ 2\ -\ S \end{aligned} \qquad (4)$$

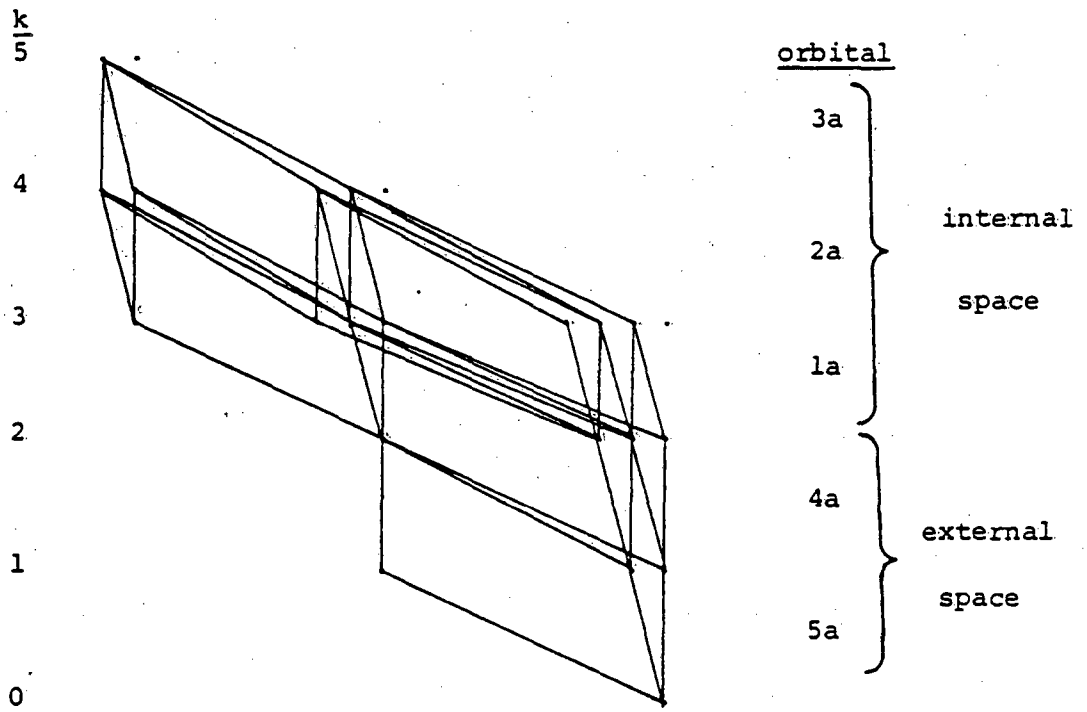
This Shavitt graph, as such a graph is called, represents all possible configurations with 5 electrons in 5 orbitals coupled to a doublet, and would therefore be used for a full-CI calculation.

Unfortunately, the size of a full CI calculation rapidly becomes prohibitive for even quite small systems. For example, one of the largest CI calculations performed to date was a full CI for water within a double-zeta (DZ) basis.[12f] For this small molecule with only 10 electrons in 14 orbitals and exploiting fully the symmetry, there are still 256 473 configurations in the full CI for the ground state ($^1A_1$). The common approach to this problem is to truncate the list of configurations to include only those configurations which are formed by a single or double replacement from one or a few references. In the case of the water calculation mentioned above, such a CI with singles and double excitations from the

Hartree-Fock reference (CISD) comprises only 361 configurations. The justification of such a treatment is based on the expectation that higher than two-body effects are not very important; indeed, in the water case, the CISD recovers 94.7% of the total correlation energy which is obtained from the much larger full-CI.

Truncation of the configuration list to certain levels of excitations from one or several references is a simple matter in the GUGA, being a matter of eliminating vertices from the Shavitt graph along with all arcs passing through the eliminated vertices. Figure 4 gives the graph appropriate for a CISD calculation including all singles and doubles from the single reference $1a^2 2a^2 3a$.

Figure 4. Shavitt graph for CISD from $1a^2 2a^2 3a$ reference.



This graph actually contains some walks corresponding to triple excitations from the reference because it was constructed considering only the number of electrons in orbitals and not the spin coupling. Suffice it to say that if such "noninteracting" configurations[15] are not desired, they can be removed from the Shavitt graph in many cases. However, this procedure would introduce considerable additional complexity to the graph in Figure 4, which would only serve to confuse the issue. Currently, Shavitt graphs can be prepared for almost all types of calculations. The most noticeable exception is that the UGA loses most of its efficiency when applied to calculations containing arbitrary sets of configurations, rather than classes, such as singles and doubles. Thus the UGA has not been applied to methods based on selecting configurations according to, for example, a perturbation theory estimate of their importance.

Although Shavitt graphs are a compact and easily visualized representation of the CI expansion, they must be reduced to a numerical form for the use of the computer. This is accomplished in the following fashion. Each vertex is numbered using a pair of indices -- the level k and a vertex number j within that level. Customarily, the vertices within a level are numbered from left to right across the graph. Each vertex corresponds to distinct row (a b c) of a Paldus tableau and so is referred to as a distinct row. The list of all such rows in a graph along with various numbering and chaining indices is called a distinct row table (DRT).

Table 2 presents the distinct row table corresponding to Figure 4.

Table 2. DRT for CISD from $1a^2 2a^2 3a^1 4a^0 5a^0$.

I	J	a	b	c	J_0	J_1	J_2	J_3	Y_0	Y_1	Y_2	Y_3	$\underline{x}$	K_0	K_1	K_2	K_3	Z_0	Z_1	Z_2	Z_3	$\bar{x}$
5	1	2	1	2	1	2	3	4	0	10	25	36	54	-	-	-	-	-	-	-	-	1
4	1	2	1	1	-	1	2	3	-	0	3	4	10	1	-	-	-	0	-	-	-	1
	2	2	0	2	1	-	3	4	0	-	3	9	15	-	1	-	-	-	0	-	-	1
	3	1	2	1	2	3	5	6	0	1	7	8	11	-	-	1	-	-	-	0	-	1
	4	1	1	2	3	4	6	7	0	6	12	15	18	-	-	-	1	-	-	-	0	1
3	1	2	0	1	-	-	-	1	-	-	-	0	3	2	1	-	-	0	1	-	-	2
	2	1	2	0	-	-	-	2	-	-	-	0	1	3	-	1	-	0	-	1	-	2
	3	1	1	1	-	1	2	3	-	0	3	4	6	4	3	2	1	0	1	2	3	4
	4	1	0	2	1	-	3	4	0	-	3	5	6	-	4	-	2	-	0	-	1	2
	5	0	3	0	-	2	-	-	-	0	-	-	1	-	-	3	-	-	-	0	-	1
	6	0	2	1	2	3	-	-	0	1	-	-	3	-	-	4	3	-	-	0	1	2
	7	0	1	2	3	4	-	-	0	2	-	-	3	-	-	-	4	-	-	-	0	1
2	1	1	0	1	1	-	2	3	0	-	1	2	3	4	3	-	1	0	2	-	6	8
	2	0	2	0	-	2	-	-	-	0	-	-	1	6	5	3	2	0	2	3	7	9
	3	0	1	1	2	3	-	-	0	1	-	-	2	7	6	4	3	0	1	3	5	9
	4	0	0	2	3	-	-	-	0	-	-	-	1	-	7	-	4	-	0	-	1	3
1	1	1	0	0	-	-	-	1	-	-	-	0	1	1	-	-	-	0	-	-	-	8
	2	0	1	0	-	1	-	-	-	0	-	-	1	3	2	1	-	0	9	18	-	26
	3	0	0	1	1	-	-	-	0	-	-	-	1	4	3	-	1	0	3	-	12	20
0	1	0	0	0	-	-	-	-	-	-	-	-	1	3	2	-	1	0	20	-	46	54

As mentioned before, the column labeled by I gives the level within the graph and the column labeled J numbers the distinct rows within a level. The columns a, b, and c give the row of the Paldus tableau associated with each distinct row. The following three sets of entries, J, Y, and $\underline{x}$ are the chaining indices, arc weights and vertex weights for what is called lexical order. The chaining indices J_s indicate which vertex in the next level down the graph that an arc of case value s connects to; whereas the arc Y_s gives the weight of the arc of case value s. In both of these arrays, invalid entries -- arcs which do not exist in the Shavitt graph -- are denoted by a dash. The vertex weight $\underline{x}$ is the number of lower walks from a vertex, or equivalently the number of different paths from the vertex to the graph tail. The lexical number, m, of a walk or configuration is given by the sum of the arc weights for that walk:

$$m = 1 + \sum_{i=1}^{n} Y_{si} \qquad (5)$$

where the extra subscript i refers to the level in the graph.

This lexical numbering scheme is one of the most powerful aspects of the GUGA. The configurations are rationally numbered from one to the number of configurations with no duplication of numbers or gaps in the sequence. Furthermore, the lexical number of a configuration is rapidly and easily determined by the sum in Equation 5. Lexical order is a downward directed scheme and so is useful when finding the configurations which are identical from the graph head to a

particular vertex but then differ between the vertex and the graph tail. This set of configurations share a common upper partial walk but different lower partial walks from the vertex in question. Letting the sum of the arc weights over the upper walk be m_u, then there will be $\underline{x}$ such configurations and due to the nature of lexical order, their lexical numbers will be m_u+1, m_u+2, ..., $m_u+\underline{x}$. For this reason, the vertex weight $\underline{x}$ will be referred to as the number of lower walks from a vertex. That these lower walks from a given upper partial walk are a sequential set of configurations will be most useful.

For the set of configurations sharing a common lower partial walk but different upper partial walks, there is no such relationship within lexical order. Their configuration numbers will be essentially randomly spaced throughout the list of configurations. However, in reverse lexical order, the upward directed counterpart of lexical order, these configurations sharing a partial lower walk would again be a sequentially numbered set of configurations. The last three sets of columns in Table 1 (K_s, Z_s, and $\bar{x}$) are the chaining indices, arc weights and vertex weights, or number of upper walks, for the reverse lexical order. The reverse lexical number of a configuration is given by Equation 3 with Y replaced by Z.

Unfortunately, we are left with two distinct numbering schemes, one of which is useful for lower walks and the other for upper walks. One approach to this dichotomy is the use of an indexing array A, as proposed by Brooks.[15c,d] Given a lexical configuration number m,

then A(m) is the reverse lexical number of the configuration. There are several disadvantages to this rather direct approach to the problem, not the least of which is that the indexing vector is the length of the configuration list.

B. Matrix Elements

The discussion so far has centered on the generation of the Shavitt graphs and the corresponding distinct row tables, as well as on the two numbering schemes for the configurations. The next section will present an outline of the methods used to evaluate Hamiltonian matrix elements during the CI calculation. The derivation of the formulae presented will not be discussed; again, the reader is referred to the articles in particular of Paldus[14a] and Shavitt[14b] for further details.

The spin-independent electronic Hamiltonian that will be used, in its second-quantized form, is

$$H = \sum_{ij} \langle i|h|j\rangle \sum_s X^+_{is} X_{js} + \tfrac{1}{2} \sum_{ijkl} [ij;kl] \sum_{rs} X^+_{ir} X^+_{ks} X_{jr} X_{ls} \qquad (6)$$

where X^+_i and X_i are the creation and annihilation operators for spinorbital $|is\rangle$, and $\langle i|h|j\rangle$ and $[ij;kl]$ are the one and two electron integrals over molecular orbitals, respectively. introducing the following one and two body unitary group operators:

$$E_{ij} = \sum_s X^+_{is} X_{js} \tag{7}$$

$$e_{ij,kl} = E_{ij}E_{kl} - \delta_{kj}E_{il} = e_{kl,ij} \tag{8}$$

then Equation 6 in the formalism of the UGA becomes

$$H = \sum_{ij} \langle i|h|j\rangle E_{ij} + \tfrac{1}{2} \sum_{ijkl} [ij;kl] e_{ij,kl} \tag{9}$$

The matrix element $H_{m'm}$ of the Hamiltonian between two configurations or Gelfand states m' and m is

$$H_{m'm} = \langle m'|H|m\rangle = \sum_{ij} \langle i|h|j\rangle \langle m'|E_{ij}|m\rangle + \tfrac{1}{2} \sum_{ijkl} [ij;kl] \langle m'|e_{ij,kl}|m\rangle \tag{10}$$

At this point, the relationship between the matrix elements of the UGA operators and the coupling coefficients of the direct CI method[11] is obvious:

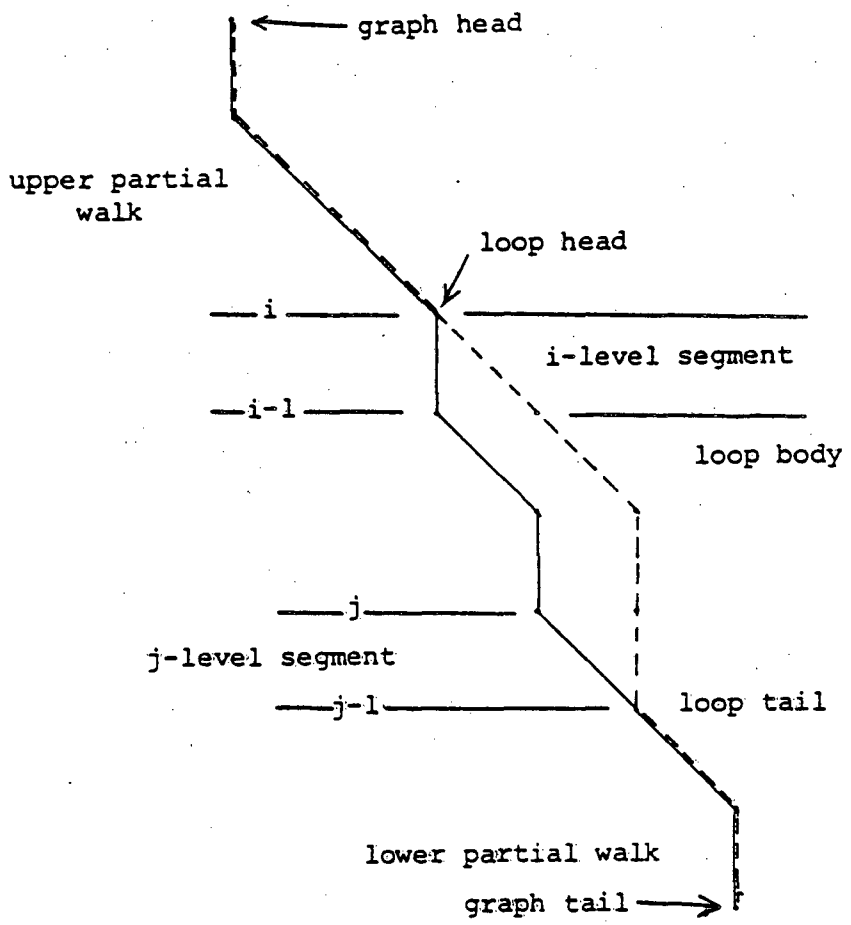
$$A^{m'm}_{ij} = \langle m'|E_{ij}|m\rangle \tag{11}$$

$$A^{m'm}_{ijkl} = \tfrac{1}{2} \langle m'|e_{ij,kl}|m\rangle \tag{12}$$

Given a configuration list and the corresponding DRT, the task of constructing the Hamiltonian matrix reduces to finding the matrix elements of the unitary group operators and using these as the coefficients for the integrals as in the direct CI approach. Graphically, the matrix elements of the unitary group operators correspond to loops between walks on the Shavitt graph, such as

in Figure 5.

Figure 5. Loop corresponding to the matrix element $\langle m' | E_{ij} | m \rangle$



The loop depicted in Figure 5 is a one electron loop determining the coupling coefficient $A_{ij}^{m'm}$ of the integral $\langle i|h|j\rangle$ in the matrix element $H_{m'm}$. The value of the coupling coefficient, which is often simply called the loop coefficient, is a function of only the shape of the loop and its lateral position in the Shavitt graph. Specifically, it is a product of segment values over the range of the loop:

$$A_{ij}^{m'm} = \langle m'|E_{ij}|m\rangle = \prod_{k=i} W(T_k, b_k) \qquad (13)$$

T_k is the segment shape symbol and b_k the b value of the ket m at the k^{th} level of the graph.

The matrix elements of the two body operator are somewhat more complicated but fundamentally the same concept applies. Noting from Equation 8 that $e_{ij,kl}$ can be expressed in terms of one body operators, one has

$$\langle m'|e_{ij,kl}|m\rangle = \langle m'|E_{ij}E_{kl}|m\rangle - \langle m'|E_{il}|m\rangle \delta_{kj} \qquad (14)$$

The first term can be rewritten using a summation over intermediate states $|m''\rangle$ to give the following:

$$\langle m'|e_{ij,kl}|m\rangle = \sum \langle m'|E_{ij}|m''\rangle \langle m''|E_{kl}|m\rangle - \langle m'|E_{il}|m\rangle \delta_{kj} \qquad (15)$$

In practice, the summation over intermediate states is inconvenient;

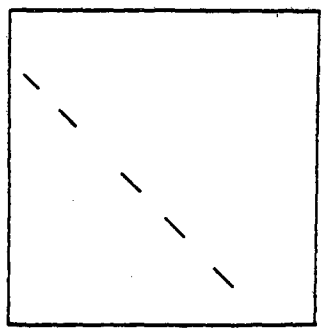
however, Drake and Schlesinger,[14c] Paldus and Boyle,[14d] and Shavitt[14e] suggested the following factorization scheme, reminiscent in the form of the one body Equation 13:

$$\langle m'|e_{ij,kl}|m\rangle = \left\{\prod_{k\epsilon S_1} W(T_k,b_k)\right\} \sum_{x=0,1} \left\{\prod_{k\epsilon S_2} W_x(T_k,b_k)\right\} \tag{16}$$

where S_2 is the range where i$\longrightarrow$j overlaps with k$\longrightarrow$l and S_1 is the range of the loop where they do not. The summation over x corresponds to singlet and triplet recouplings of the intermediate states in Equation 14, but the intermediate states do not explicitly appear in this formulation.

Since the loop value depends only on the shape of the loop and not on the upper and lower walks from the loop to the graph head and tail, a loop can determine the value of a series of coupling coefficients differing only in m' and m. If there are $\underline{x}$ lower walks from the loop and $\bar{x}$ upper walks, the loop will contribute a value $A_{ij,kl}[ij;kl]$ to $\bar{x}$ irregularly spaced groups of $\underline{x}$ sequential matrix elements along an off-diagonal of the Hamiltonian matrix (provided we use lexical order). This is illustrated in Figure 6.

Figure 6. Locations in H matrix of contribution $A_{ijkl}[ij;kl]$ from loop with $\underline{x}$ lower and 5 upper walks.



In summary, the UGA provides a spin-adapted orthonormal set of n-electron functions which serve as the configurations for the expansion of the CI wavefunction. These configurations are readily limited to only certain classes of configurations, such as all singles and doubles from a set of reference configurations. Furthermore, the list of configurations is easily ordered and numbered using either lexical or reverse lexical order. The number of any particular configuration is found as a sum of arc weights. The coefficients of the one and two electron integrals in Hamiltonian matrix elements -- the coupling coefficients -- are related to the matrix elements of the one and two body unitary group operators, which, in turn, are determined by the shape and location of loops in the Shavitt graph. The value of a one-electron loop is a simple product of segment shape values; that of a two electron loop, a sum over singlet and triplet recoupled terms, each of which is a simple product of segment shape values.

III. PREVIOUS IMPLEMENTATIONS

The first implementation of a CI program based on the GUGA was that of Brooks and Schaefer[15c] in 1978. A preliminary version of the program produced a formula tape from which a diagonalization tape was constructed. The formula tape contained the coupling coefficients, information to identify the integrals, the "primary" configurations involved, and the number of upper and lower walks from the loop. The primary configurations and number of upper and lower walks is sufficient data to determine all Hamiltonian elements to which the current loop contributes. The program then combined the formula tape and the integrals to form a diagonalization tape containing an actual value of a loop contribution as well as the configurations and number of upper and lower walks.

For a CI based on the Davidson algorithm[16] for finding the lowest root or few roots of the Hamiltonian, there is no need to actually form the Hamiltonian matrix. The diagonalization tape described above is sufficient. What must be computed during each iteration of the Davidson scheme is the correction vector Z:

$$Z_I = \sum_J H_{IJ} C_J \qquad (17)$$

A loop contribution $A^{IJ}_{ijkl}[ij;kl]$ is not necessarily the same as the Hamiltonian matrix element H_{IJ}, although they often are identical. In particular, if the configurations I and J are either the same or a single excitation from each other, the matrix element will be the

sum of many different loop contributions. In Equation 17, the contributions may be summed to form H_{IJ} multiplied by C_J and summed into Z_I, or each contribution can be multiplied by C_J and summed. The importance of the latter approach is that the various contributions to a particular Hamiltonian matrix element tend to be located throughout the diagonalization tape, and it would require a very extensive and difficult sort to combine them to form the actual Hamiltonian matrix.

The preliminary program also took advantage of the fact that the Hamiltonian matrix is symmetric. Thus, it is necessary only to form contributions to matrix elements such that I J, and use the following equation as well as 17 to account for the cases where:

$$Z_J = \sum_I H_{IJ} C_I \qquad (18)$$

A loop contribution $A^{IJ}_{ijkl}[ij;kl]$ is therefore multiplied by C_I and added to Z_J and by C_J and added to Z_I.

The remaining problem is to determine the values of I and J from the primary configuration numbers I' and J' as well as the numbers of upper and lower walks $\bar{x}$ and $\underline{x}$. This was solved by Brooks, who called the solution the loop breakdown algorithm. Using an indexing array A such that A(i) is the reverse lexical configuration number of the lexical configuration i, the loop

contributes to all H_{IJ} such that

$$\begin{aligned} I &= A(I'+i) + j \\ J &= A(J'+i) + j \end{aligned} \qquad \text{for} \begin{cases} i = 1,2,3, \ldots, \underline{x} \\ j = 1,2,3, \ldots, \bar{x} \end{cases} \tag{19}$$

Note that I and J are reverse lexical configuration numbers.

The processing of the diagonalization tape is seen to be relatively simple and only requires that the three vectors C, Z, and A be randomly accessible. For the moment, let us not dwell on these requirements but rather investigate how the formula tape was constructed in the first place. The task is to find and evaluate all possible loops using the Shavitt graph as a template. The algorithm used -- the loop driven approach[15c] -- is central to the generality and efficiency of the current UGA based programs. The algorithm consists of the following recursive search.

Beginning at the top level of the graph, the search starts by finding a valid opening segment for a loop. The search then continues with valid body segments of the loop, keeping to the left as much as possible. As a new segment is accepted, the two arc weights associated with the bra and ket walks are added to the configuration numbers I' and J' and stored in a pushdown stack. The segment shape value (W of Equation 16) is multiplied by the current loop value and also stored in a pushdown stack. When a loop is completed, the loop value, integral indices, primary walks I' and J', and the number of upper and lower walks are written to

the formula tape. The search then continues at that same level but more to the right on the Shavitt graph. When no more valid segments are found, all the stacks are "popped" and the search continues one level back toward the top of the graph, but with segments more to the right on the graph. Eventually the search returns to the initial level and finds another opening segment, and finally exhausts all possible opening segments. At this time, all loops opening at the initial level of the search have been found and evaluated. By initiating the search at all levels of the graph, all loops contained within the graph are evaluated and the formula tape will contain sufficient information to construct the Hamiltonian matrix, or, equivalently, to use the Davidson diagonalization technique.

The Shavitt graph, in its numerical form as a DRT, is used only as a template for the loop-driven search. The actual search is controlled by the loop searching master table, a copy of which is found in the appendices. The master table is a list of all the possible segment shapes, their values, and implicit in the location of segments in the table, the way in which segments may be validly connected to form loops. This table is completely general, so provided the DRT can be generated, the Hamiltonian matrix can be evaluated using the loop-driven algorithm. This generality is one of the important attributes of the UGA, since almost any type of calculation can be handled.

Another aspect of the loop-driven approach is that each loop is evaluated with very little work. Though a loop may be the product of say 50 segment values, the recursive nature of the search means that each successive loop value is determined by one or two segment values in addition to the partial product shared with the last loop evaluated. As a result, the loop-driven approach is reasonably efficient in terms of computer time.

This preliminary program was remarkably successful and provided compelling evidence that the UGA would provide the basis for a new generation of more powerful CI programs. Two criteria are of practical importance when considering a CI program: first, can the program handle the type and size of calculation desired? Is it general enough? And second, how long in the sense of computer time does a given calculation take? The importance of execution time arises because a typical CI calculation might take several hours of central processor time (CPU time), and is therefore quite expensive. A factor as small as two in execution time may have a significant effect on the utility of a program.

It is in this light that Brooks' program was successful. The program could handle closed and high-spin open shell references as well as open shell singlets and two-configuration reference functions. The CI expansion could be truncated at an arbitrary excitation level from the reference(s), e.g. at singles, doubles, and triples. Furthermore, for a single high-spin open shell reference, the configuration list could be restricted to the

Hartree-Fock interacting configurations. A few more exotic calculations were undertaken, such as a three open-shell reference calculation on C_2H_4,[15d] and it is fair to say that any restrictions in the type of calculations that the program could handle were largely due to a lack of demand for such calculations rather than an inherent defect in the method. This flexibility is a hallmark of the UGA CI programs.

The score on the second criterion was perhaps the most impressive aspect of this initial program. Written in a few months and based on the incompletely understood UGA, the program nevertheless compared favorably with conventional CI programs which had been perfected and honed over perhaps ten years. For instance, the GUGA program was nearly four times faster than a conventional CI program for a 2355 configuration calculation on BH_3 within the C_{2v} symmetry subgroup.[15d]

Subsequently, various improvements were made to the preliminary program and the final version of the program does the same calculation in one sixth the time required by the conventional program. The major changes made were to directly generate the diagonalization tape without benefit of a formula tape and to avoid the summation over intermediate states in Equation 15 by using the factorization of Equation 16. This final version of the loop-driven graphical unitary group (LDGUGA) program has since been used for many calculations ranging from usual CISD (singles and doubles) from one reference up to CISDTQ calculations

on water[12e] and full CI's in small orbital spaces.

Since these programs are the direct antecedants of the program described later in this work, it will be appropriate to discuss the limitations of the LDGUGA programs as this will explain the motivation behind the shape-driven (SDGUGA) approach. Concentrating on the failures of the LDGUGA programs with the advantage of hindsight is done for pedagogical reasons and is not intended to belittle the achievement of Brooks and his coworkers.

As previously mentioned in the section about the loop breakdown algorithm, the simultaneous use of both upper and lower walks from loops requires random access to three vectors spanning the length of the configuration list. These vectors are the CI vector for the current iteration of the diagonalization, the correction vector being formed, and the indexing vector. These three vectors must be in the central memory of the computer if the unmodified diagonalization tape is used, and therefore the length of the configuration list can be no longer than about one third of the available central memory. The first implementation of the LDGUGA programs was on a Harris 6024/4 with about 32k words of central memory, so the limitation was to about 8000 configurations after allowing space for the program and incidental storage. By sorting the diagonalization tape, it is possible to do calculations up to about three times this size; however, the sorting of the diagonalization tape is a difficult and time-consuming task. This approach was used on the Slash Four minicomputer to handle

calculations with up to about 23 000 configurations,[15d] but has not been pursued since then because current computers, with larger central memories, are not so limited in this respect.

A more important limitation now is the size of the diagonalization tape itself. The tape has a somewhat more compact form than the Hamiltonian matrix itself due to the use of the number of upper and lower walks from each loop, but for a calculation with about 20 000 configurations, the tape can easily be as large as two hundred Mbytes. This amount of data is about the limit of the available mass storage space available to any one user on most computer systems, so larger calculations are impractical especially since the size of the diagonalization tape grows as the square of the number of configurations.

One final limitation on the size of calculation feasible is the time taken for the calculation. Though this is not such a definite limit as the others discussed, there is a tendency to avoid calculations taking more than a few hours to complete. In this sense, the limitations on the Harris Slash Four were perfectly matched since a 23 000 configuration calculation took from 5-10 hours.

The first step towards a method capable of very large CI calculations is to eliminate the diagonalization tape. The LDGUGA programs discussed so far have divided the calculations into two parts: the generation of the diagonalization tape and its subsequent processing each of the iterations of the Davidson

algorithm. A simple solution is to combine the two phases of the calculation and generate and use the loops each iteration of the calculation. Certainly this eliminates the lengthy diagonalization tape; unfortunately, this tack results in a rather slow program. When the loops are evaluated but once and then stored, it is not important that the generation of the loops takes about three times as long as each subsequent iteration of the Davidson scheme because there are typically 7-8 such iterations. But generating the loops each iteration, the calculation would take about three times longer.

Certainly, a slow program that can do the calculation is preferable to a fast one that cannot, but it is not an appealing solution. Also, the limit of three vectors in core still holds, which on the newer Harris-800 minicomputer results in a maximum CI size of perhaps 90 000 configurations. The three vectors need to be simultaneously in the central memory because of the loop breakdown algorithm's use of both upper and lower walks of a loop. As mentioned in the description of the theory of the UGA, using both the lower and upper walks must result in large changes in the configuration numbers no matter whether lexical or reverse lexical order is used. The solution is to use either the upper walks and reverse lexical order or the lower walks and lexical order, but not both. Either approach results in recalculating some loops a number of times and would result in the already slow program becoming even slower. Given these thoughts, it is obvious that

a faster method for evaluating loops is needed. The shape-driven approach was designed as a solution to this problem.

IV. SHAPE-DRIVEN METHODOLOGY

The shape-driven approach was originally designed for calculations where no more than two-electrons are allowed into a set of virtual orbitals in any of the configurations in the CI expansion. This includes such classes of calculations as singles and doubles from one or a few references (CISD), first-order and second-order CI's. For the present, the discussion will exclude calculations such as CI with singles, doubles, triples and quadruples (CISDTQ) or full-CI's. Extensions of the original concepts to include such exotic calculations will be discussed later. Therefore, the following sections will presume that the program is intended for CISD type calculations where there are a substantial number of virtual orbitals in the external space.

Siegbahn noted[17] that for such a calculation, the external portion of the Shavitt graph, that is, those orbitals unoccupied in all the reference configurations, is extremely simple and quite repetitive if placed at the bottom of the Shavitt graph. It is possible to exploit this simplicity to determine a priori the value of the portion of any loop in this external space and thereby avoid the need to multiply segment shape values together for the virtual orbitals. The product in Equation 16 is factored into an internal and external portion. The internal portion is still found as a product of segment shape values, but the external part is explicitly known and incorporated in the code. The program Siegbahn developed using this explicit treatment of the external space first generates

all the internal portions of loops and writes them out on a disc file. In the second phase, the internal portions of loops are linked to the appropriate external values using an integral-driven scheme overall.

The shape-driven approach is based on the same concept of the explicit evaluatation of the external portions of loops; however, unlike Siegbahn's program, the shape-driven approach strives to take advantage of the structure inherent in the UGA. The name "shape-driven" arises from the nature of the algorithm used. For a given internal part of a loop, all external portions of a given shape are treated before processing loops of a different shape. In a sense, loops are grouped according to the value of the coupling coefficient.

The current approach is closely related to that used by Shavitt's group. The concept of using external shapes was first introduced by Shavitt[14e] in 1979, although no program based on the external shapes was developed at that time. A version of the LDGUGA program was developed in the Schaefer group which was a direct CI program based on the external shapes of loops presented by Shavitt. However, the method continued to use a loop-driven metholology and therefore was not able to as fully exploit the structure of the UGA as the current program. Apparently, the program developed by Lischka, et al.[18] does exploit the external shapes in much the same way that the current method does, although it is not yet known how closely the two algorithms resemble each other.

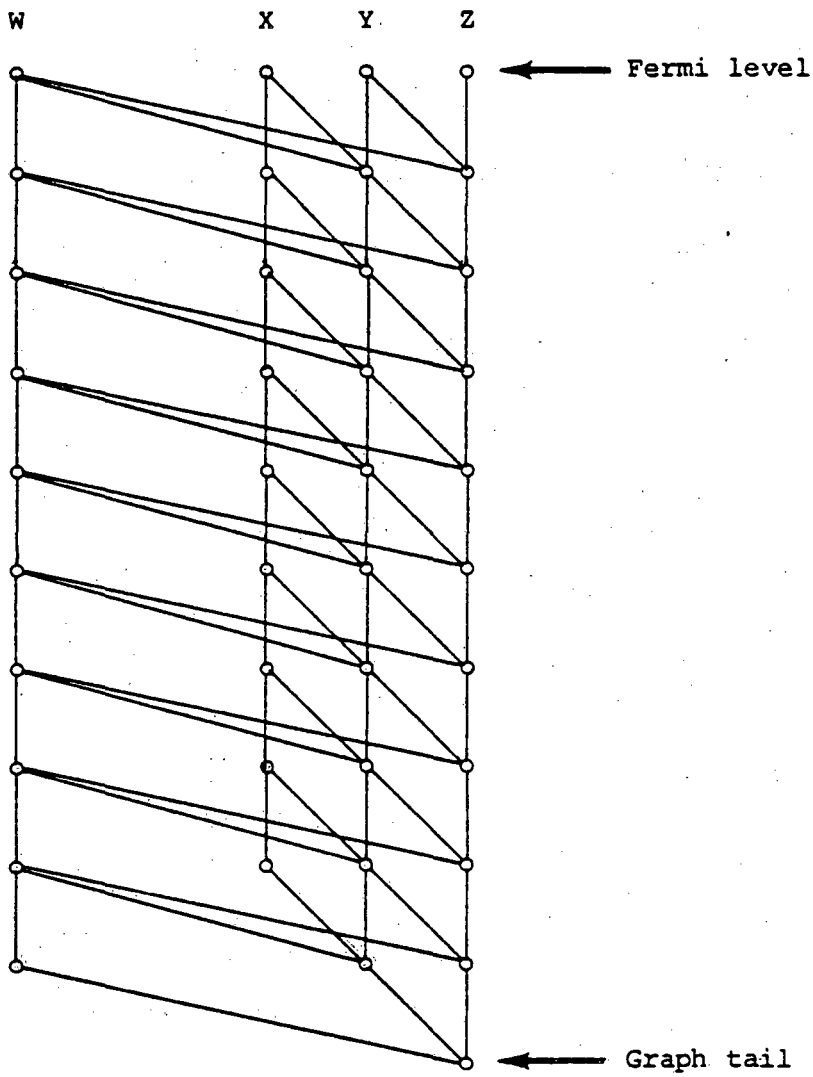
What are the advantages of this new approach, which appears so similar to Siegbahn's? The principal advantage lies in being able

to visualize the problem and therefore be able to break the program into small, easily understood, and easily optimised segments. Furthermore, by not ignoring the structure of the CI calculation, it will become apparent how to undertake large CI calculations where it is impossible to simultaneously hold both the entire CI vector and correction vector in the central memory of the computer. Returning for a moment to the segmentation of the computer code, by specializing each segment of code to just one shape of external portion of loop, it is possible to eliminate almost all decisions from the code. Progressing through a set of loops of the same shape, the configuration numbers I and J can be found by incrementing counters rather than by a complicated and time-consuming look-up. Finally, since each segment of code is quite short and deals with only one procedure, it is relatively simple to optimize the program.

The following specific example will illustrate most of the concepts used in the shape-driven approach. Before embarking on this discussion, however, it is necessary to explain the terminology that will be used. Figure 7 shows the external space for the type of calculation we are interested in at the present. The modified Fermi level is that level which separates the internal and external portions of the Shavitt graph, ie. none of the orbitals below the Fermi level is occupied in any of the references, and so will often be referred to as virtual orbitals.

For the external space shown, corresponding to no more than two electrons in the external space and no molecular symmetry present,

Figure 7. External space of Shavitt graph for CI singles and doubles, illustrating the simplicity and regularity.



there are no more than four distinct rows (vertices) at any level. These rows, and in particular the four at the Fermi level, are labeled W, X, Y, and Z when there are two electrons singlet or triplet coupled, one electron, or no electrons in the remaining levels. An alternative notation used by Seigbahn[17] is S, T, D, and V, standing for singlet, triplet, doublet and valence, respectively.

Figure 8 shows a typical internal portion of a loop. Associated with this partial loop are the partial values of the configuration numbers I^i and J^i and of the loop coefficients A^i and B^i. There are also the appropriate part of the integral address (ARR) and symmetry information connected with the integral addressing (ASM). Finally, the loop arrives at the Fermi level as an XX entry to the external routines and the next segment shape must come from the ISEG=8 section of the loop-searching master table. This loop is termed an XX entry because both the I and J partial walks arrive at the Fermi level at the X point. The final piece of data is that the TRACK value of the internal loop is 3. The TRACK value determines which of a triplet of integrals to use when forming the loop contribution. Table 3 illustrates the use of the loop-searching master table in Appendix 1 to evaluate loop coefficients and track value for this loop.

That the sample loop is an XX entry to the external space with an ISEG value of 8 determines that the eight shapes of external portion of the loop shown in Figure 9 along with a notation indicating the coefficient value and integrals to use in

Figure 8. Internal portion of a loop.

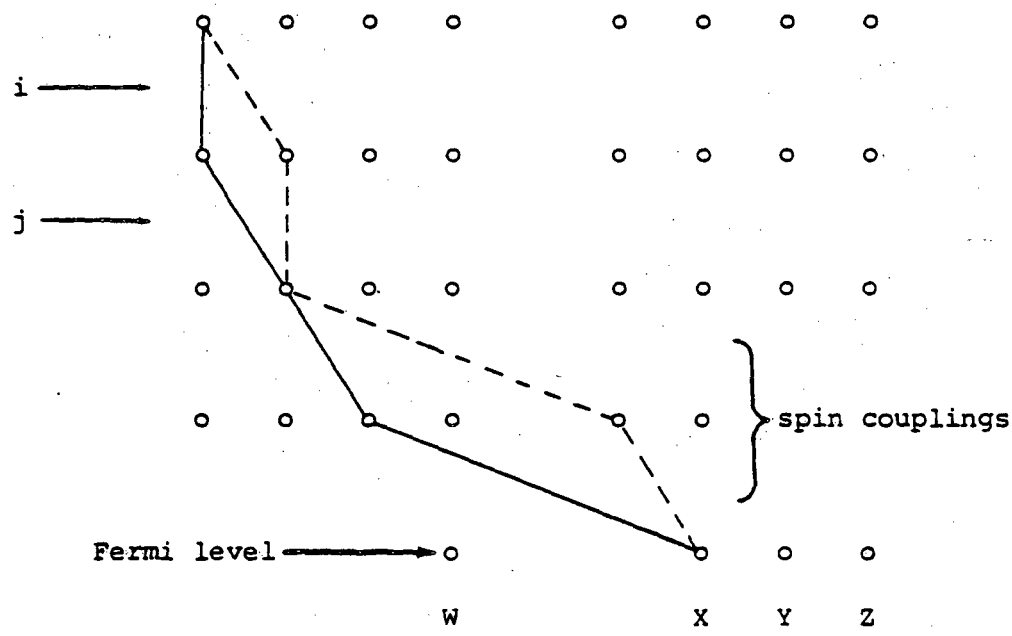
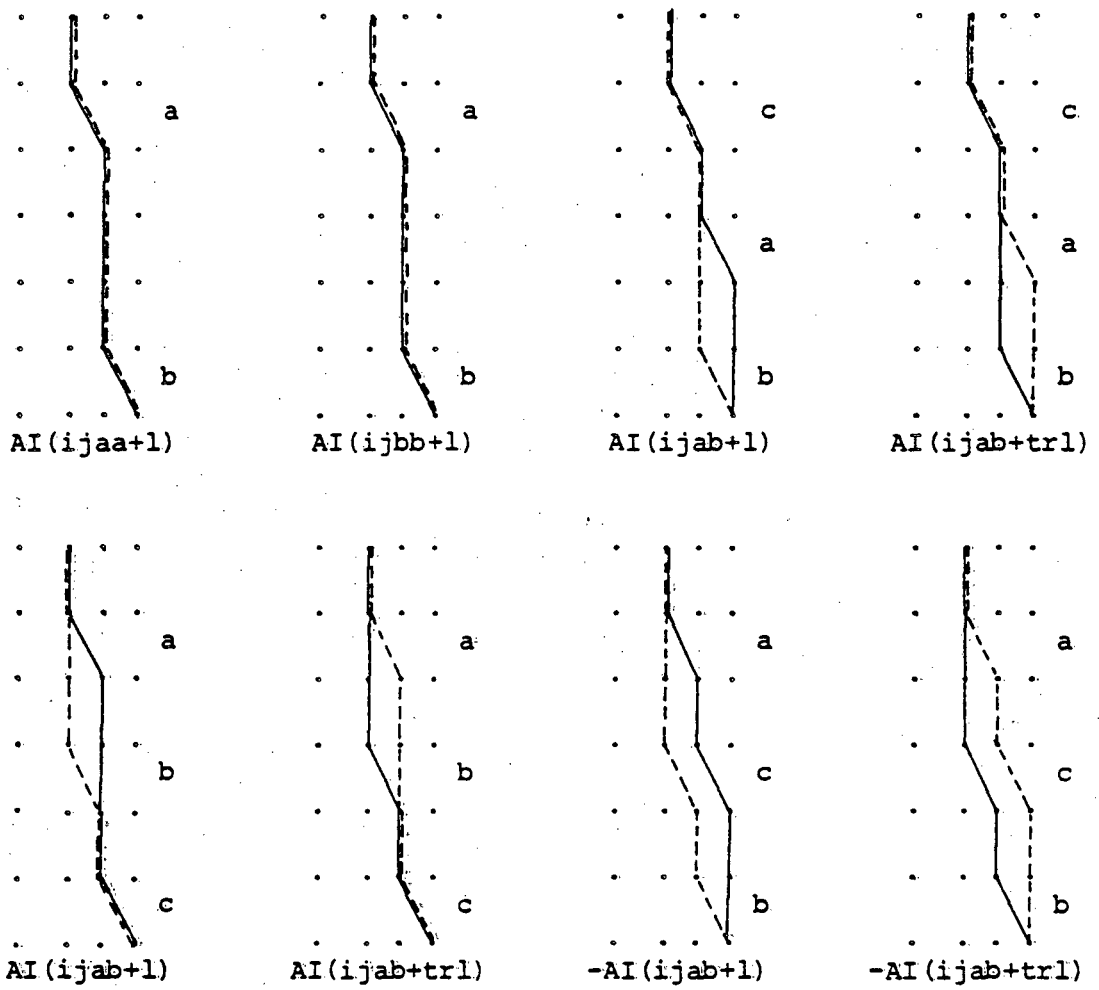


Table 3. Evaluation of the loop coefficients for the internal portion of the loop shown in Figure 8. Refer to Appendix 1 for an explanation of the entries in this table and for the loop-searching master table.

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	VALUE	b	X	Y	Comments
1	1	3	01	1	-10	$X' = X$	3	1	-	i-level of loop is this level. JCOND of -1 indicates that j-level not yet reached.
3	38	6	10	1,2	10	$X' = X\sqrt{\frac{1}{2}}$ $Y' = -X\sqrt{\frac{b}{2(b+2)}}$	2	$\sqrt{\frac{1}{2}}$	$-\frac{1}{2}$	JCOND of 1 indicates that j-level of loop is this level.
6	82	7	12	3	00	$X' = -Y\sqrt{\frac{2}{b(b+1)}}$	2	$\sqrt{\frac{1}{12}}$	-	KCOND of 0 indicates this is not k-level.
7	98	8	21	-	00	$X' = -\sqrt{2}X/b$	3	$-\sqrt{\frac{2}{27}}$	-	Still not k-level. TRACK remains set to 3.

Figure 9. The eight possible external shapes for the internal portion of the loop shown in Figure 8. Note that A is called X in Table 3.



forming the loop contribution are the only possible continuations from the internal portion of the loop. In this example, the external loop coefficient is ±1 so the total loop coefficient is $\pm A^i$, as reflected in the formulae of Figure 9. Note that the second loop coefficient B is not used in this example and that for some shapes the track value from the internal space is superceded by an external value.

To further explain how the shapes lead to segments of the computer program, the following section will examine the last shape listed in Figure 9 in detail and derive from the information in the Figure a prototype of the Fortran code used in the program. The external portion of the bra walks configuration number is determined by the levels c and b since vertical arcs make no contribution to configuration numbers. In general, it would be necessary to define an array WTX(c) which contains the arc-weight of the arc from the X point at level c to the Y point at level c-1. In the present example, this will turn out to be unnecessary, but for the time being, the total configuration number I of the bra walk is given by I^i + WTX(c) + b. There is no need for an array analogous to WTX for the b level contribution since within lexical order all lower walks from the Y point arrived at by the arc from the X point at level c are contiguous and numbered in a leftmost, that is bottom to top, sense. In a similar fashion, the configuration number J of the ket walk is J^i + WTX(a) + c. The integral address pointer needed is ARR + KADD(a) + LADD(b), where the first term has been evaluated in

Figure 10. FORTRAN code generated to handle the last external shape of Figure 9.

```
      VALA = -A^i
      IA = J^i
      DO 3 A=3,LEVFRM
        IA = I^i
        IJA = ARR + KADD(A)
        DO 2 C=2,A-1
          CJVALA = C(JA) * VALA
          ZJ = 0.0
          DO 1 B=1,C-1
            TEMP = INTGRL(IJA+LADD(B)+TRACK)
            Z(IA) = Z(IA) + TEMP *CJVALA
            ZJ = ZJ + TEMP * C(IA)
            IA = IA + 1
    1     CONTINUE
          Z(JA) = Z(JA) + ZJ * VALA
          JA = JA + 1
    2   CONTINUE
    3 CONTINUE
```

the internal space. Putting this information together yields the Fortran code of Figure 10. Note that in this specific case, it was possible to dispense with the auxilliary weight array WTX because of the particular order of the nesting of the DO loops, using the fact that the lower walks from the X point at the Fermi level are a contiguous set ordered in a leftmost sense.

For every external shape, there is a section of Fortran code comparable to that in Figure 10. In all important sections of code, the constant coupling coefficient A, but not always B, can be removed from the innermost program loop since either the bra or ket configuration number is constant within the innermost loop. Because the multiplication by the loop coefficient can be removed from this innermost program loop, the construction of Hamiltonian elements is an insignificant time step when using SDGUGA! In a later section concerned with the vectorization of the code, it will be shown that by reorganizing the program somewhat, it is possible to completely remove the coupling constants from the innermost loop. In this case, it is worth emphasizing that a calculation would be essentially no faster even if the Hamiltonian matrix elements were available at no cost or could be looked up in a master list somehow. It would still be necessary to increment a configuration number or some other counter, pick up the Hamiltonian matrix element and multiply it by C_I and C_J. The current program does equivalent work in the innermost program loop, so for a moderate size calculation -- one with

perhaps 30-50 external orbitals -- the SDGUGA CI program is essentially as fast as any program can be which multiplies the CI vector by the complete Hamiltonian matrix to form a correction vector.

In principle, it should be possible to neglect Hamiltonian matrix elements which are essentially zero since they cannot significantly affect the results. A program based on such a selection of matrix elements could be considerably faster than a method such as the present one which uses all matrix elements irrespective of their magnitude. However, no such programs are currently in use, and it is possible that the process of selecting matrix elements could be a prohibitively expensive task.

V. PAGING OF THE VECTORS

Understanding the structure of the UGA and how it is being used in the shape-driven approach allows the implementation of an algorithm for handling large CI expansions, where it is impossible to simultaneously hold all elements of both the CI and correction vectors in the central memory of the computer. From the analogy with the method used to handle the virtual memory of the computer itself, this process will be called "paging" of the vector. The CI and correction vectors will be stored on a random-access disc file and only those portions actually needed at one time will be transferred on demand to the central memory. For such a scheme to be practical, the calculation must not be I/O bound, that is, a substantial amount of central processor time must elapse between successive paging operations. If this is not the case, the calculation will spend most of the time waiting for the completion of a paging operation since I/O is much slower than the central processor. For example, a basic operation such as a multiplication might take a fraction of a microsecond; paging one section of the vector into central memory, 0.05 seconds.

The shape-driven algorithm outlined in the preceding sections forms loop contributions to rectangular submatrices of the Hamiltonian matrix. Only for diagonal elements and configurations differing by only one electron are there more than one contribution per matrix element, so by and large, the Hamiltonian matrix will be

constructed as a set of rectangular submatrices of moderate dimension. In the particular example of the preceding sections, if there is no symmetry and N_e external orbitals, the submatrix constructed will be a square of dimension $N_e(N_e-1) / 2$. If N_e is say 50, the size of the submatrix is over 1000 x 1000, which is large enough that there will be a considerable amount of central processor work involved in evaluating the loop contributions (there are about 100 000 nonzero contributions to the submatrix), but not so large that the five or six thousand vector elements needed cannot readily be stored in the central memory.

The strategy used to minimize the amount of time spent on the paging of the vectors is to page in, that is, transfer the appropriate section from the peripheral storage to the central memory, as large a portion of the vectors as possible at one time, and then do as much work as possible on those sections of the vectors before paging them back out to the disc file. This is accomplished by modifying the loop-driven algorithm for the internal portions of loops so that the number of upper walks from a loop is ignored. Thus, the loop is recomputed for each upper walk, but it was seen in the last section that the shape-driven approach spends a negligible amount of effort evaluating loop coefficients, so this repeated work will not noticeably affect the performance of the program. The purpose of this modification of the loop-driven algorithm is to ensure that the paging through the vectors is done as methodically and efficiently as possible.

The loop-driven algorithm previously outlined searched across the distinct rows at a given level, finding all loops opening at one distinct row before proceeding to the next. The simultaneous use of the upper walks from the loops causes the configuration numbers I and J of the configurations making the loop to change by large amounts. Thus, when using the upper walks to avoid recomputing the value of the loop, contributions will be made to segments of the correction vector near the beginning, at various locations in the middle of the vector and also near the end, for each distinct row at that level. Therefore, it would be necessary to page through the vector once for every distinct row. This can be avoided by always searching from the graph head down to the level of interest to find the upper walks from the loops one by one, keeping a leftmost sense to the search.

Searching in this fashion, there are two different modes of paging to consider. The simplest situation is when it is possible to hold enough elements of the vectors while still descending to the level of loop opening. Remember that all the lower walks from any given distinct row are a continguous set. Then, while descending from the graph head toward the level of loop opening, it is sufficient to check the number-of-lower-walks, $\underline{x}$, from each distinct row reached. When $\underline{x}$ is less than half the available central memory, $\underline{x}$ elements of each vector, starting with the current partial configuration number, are transferred into core. The search then descends further down the graph before opening loops, and then recursively continues, but all the while the elements of the vector

needed will always be available. Only when the search returns to the level where the vector was paged into memory must it be paged back out and a new portion paged in.

This first mode of paging is extremely efficient. The vector is paged through sequentially once for each level in the graph. Indeed, in smaller calculations it is possible to pass through the vector substantially less often by holding larger numbers of integrals in the central memory. The second mode of paging occurs when the loop has already been opened before sufficient elements of the vectors can be held at once. This mode of paging is somewhat more complicated, and it is not possible to know how many times the vector must be passed through each iteration of the calculation without actually trying the calculation.

In this second case, it is not possible to accomodate the number-of-lower-walks-from-the-loop-head elements of each of the vectors. Therefore, this set of elements will have to be split into subgroups which can be held in core. Each subgroup will have to be resident in the central memory at the same time as each other subgroup, i.e. if the number-of-lower-walks segment of the vector is divided into n sections, then the first section will have to be in core with the second section, and then with the third, etc. This will require about $n^2/2$ paging operations, but if the available core space is moderately large, say 100 000 elements or more of each vector can be held, then the amount of work per page request should prevent the program from being I/O bound.

The actual implementation of the second mode of paging is similar to that used in the first. As the loop extends down the graph, the I and J walks will end on a pair of distinct rows. When the sum of the lower walks of both these points is less than half the available space, the four portions of the vectors -- two each of the CI and correction vectors, corresponding to the I and J walks -- can be transferred into the central memory. The additional complexity in this case arises when the portion of the vector associated with only one walk needs to be replaced.

These two modes of paging will usually both occur in the same calculation, the only exception being that only the first will be needed if more than about 2/3 of the total space needed is available. In larger calculations, the second mode of paging is used when evaluating loops which open near the top left portion of the graph, since that is the region where the number of lower walks from the distinct rows is largest. The first mode of paging is encountered further down the graph. The minimum required amount of central memory for the vectors is twice the sum of the two largest numbers of lower walks from points at the Fermi level. This is no more than $2N_e^2$ elements where N_e is the number of external orbitals. For even 100 virtual orbitals then, the program can run with 20 000 words of memory dedicated to the vectors, irrespective of the length of the CI expansion. Thus, holding the necessary portions of the vector in the central memory amounts to no restriction at all on the size of the CI expansion.

What then is the limitation on the length of the CI expansion the SDGUGA program can handle? Apart from the $2N_e^2$ words of central memory mentioned above, the program needs space for about $3/2N^2$ integrals, where N is the total number of orbitals, as well as a few tens of thousands of words for arrays such as those defining the DRT. In light of this small amount of central memory required, the available memory is not a practical limitation on most computers. The total amount of disc storage available to any one user might be a limitation. The current amount needed is $N^4/8$ words for the integrals and space for two vectors per iteration of the Davidson algorithm. For a calculation with 100 orbitals and 10^6 configurations taking ten iterations, about 3×10^7 words of disc storage are required. The only remaining limitation, and the one most likely to limit the size of calculation undertaken, is the time taken. On the Harris 800 minicomputer, a calculation with 10^6 configurations took just over 100 hours to complete. Even on a scalar machine as fast as the CDC 7600, the same calculation would take almost 10 hours; therefore it would seem that time is the strongest limitation on the size of the CI expansion.

VI. MATRIX FORMULATION OF THE SHAPE-DRIVEN APPROACH

The algorithm so far described is somewhat cumbersome and involves more computational work and DO-loop overhead than is strictly necessary. It has been pointed out by Saunders and van Lenthe[19] that Siegbahn's approach[17] can be reformulated into standard matrix and vector operations such as matrix multiplications. There are several advantages to such a reformulation. Aesthetically, the program is considerably simpler and more elegant, which has the practical benefits that it is easier to write and debug. Futhermore, since the computational effort is localized in a few matrix routines, the performance of the program as a whole can be optimized by attention to just a few portions of code. It is possible in the matrix formulation to remove the coupling coefficients from the innermost program loop in all important cases, thus making the construction of the Hamiltonian matrix per se an insignificant portion of most calculations.

The main motivation behind the reformulation of the algorithm, however, is to take advantage of the very specialized and powerful vector computers and array processors becoming available presently. These devices are restricted to handling rather simple operations efficiently, and were designed primarily for vector and matrix operations. Both versions of the current program have been implemented on a CRAY 1S computer, and as will be shown later, the matrix version executes more than ten times as fast as the original CRAY version. With this performance, which is perhaps 20 times better

than that possible on the fastest scalar machines currently available, it is possible to undertake calculations on molecules which have previously been considered too large for any attempt to be made at a correlated calculation. For such calculations, the full power of the SDGUGA matrix formulation will be needed -- both speed and ability to page the vectors.

The discussion of how to manipulate the various external shapes into matrix and vector products is necessarily lengthy and involved. The following sections will consider each different type of entry to the external space independently, grouping the shapes together according to the number of indices of the loop (and integrals) in the external space. Before beginning this discussion, a shorthand method of referring to configurations will be introduced.

The configurations will be labeled according to the internal partial walk, the external orbitals occupied and the spin-coupling of the external orbitals. The general scheme will be to label a configuration as C_{ab}^{iw} where the superscripts i and w give the internal partial walk label and the entry point to the external space. The subscripts a and b denote the external orbitals occupied in this particular walk, with the orbital a being the first orbital coupled in the configuration and the orbital b, the second. Thus, the orbital a will be lower than b on the Shavitt graph and will have a smaller value. Using this nomenclature, the possible configurations are C^{iz} for a configuration with no electrons in the external space, C_{a}^{iv} for one electron, and C_{ab}^{iw} and C_{ab}^{ix} for the case

with two electrons singlet or triplet coupled respectively.

A. Four-Internal Loops

These loops, which are completed in the internal space, are the simplest to treat as they naturally form a vector operation. There is no need to extend the configurations to the Fermi level as long as sufficient portions of the vectors are in the central memory. Having just elaborated on the shorthand for configurations, it does not suffice in this one case. Associated with an internal loop are the number of lower walks, $\underline{x}$, from the loop tail and the two partial configuration numbers i and j found by summing the arc weights of the bra and ket walks, respectively, from the graph head to the loop tail. The loop naturally defines two subvectors of the CI vector, $C^{i'}$ and $C^{j'}$, which are the $\underline{x}$ elements starting with C_i or C_j respectively. Thus, for example, $C^{i'}$ is the elements C_i, C_{i+1}, $C_{i+2}, \ldots, C_{i+x-1}$. If the loop contribution is v, then the following pair of equations gives the contribution to the correction vector:

$$\begin{aligned} Z^{i'} &= vC^{j'} \\ Z^{j'} &= vC^{i'} \end{aligned} \qquad (20)$$

The correction subvectors Z' are defined in the same way as the C' subvectors. One final point is that if the loop is diagonal, that is, if the bra and ket walks are identical, only one of the equations above is used. The other equation accounts for the

symmetric nature of the Hamiltonian matrix and is not appropriate for elements on the diagonal.

B. One-External Loops

The one-external loops involve integrals with one index in the external space and arise from YZ, ZY, XY, YX, WY, and YW entries to the external space. The first two cases are already in the form of a dot product and a vector times a scalar:

$$Z^{jz} = \sum_a H_a C_a^{iy}$$
$$Z_a^{iy} = H_a C^{jz} \qquad (21)$$

These are the equations for the YZ entry. Simply interchange the superscripts i and j for the ZY case. The vector H_a of loop contributions is given by the following equation for all of the entries:

$$H_a = A\ I(ijka+tr1) + B\ I(ijka+tr2) \qquad (22)$$

A and B are the loop coefficients and I(ijka+trl) is the integral located in the triplet with the address for indices i, j, k, and a, and an offset within the triplet of trl. Thus, the integral is one of [ij;ka], [ik;ja], or [ia;jk]. An advantage of Brooks' integral storage scheme is that the integrals needed to make sequential elements of H_a are in sequential triplets and thus form a vector themselves.

The situation with the remaining four classes of entries is not so simple. Figure 11 shows the possible external shapes for the XY entries and gives the associated loop values. The evaluation of the contributions from both of the shapes can be combined into one process in the following fashion. A vector of loop contributions H is formed as before as well as an antisymmetric matrix of CI coefficients C' defined as follows:

$$C'^{i}_{ab} = \begin{cases} C^{ix}_{ab}, & a < b \\ -C^{ix}_{ba}, & a > b \\ 0, & a = b \end{cases} \qquad (23)$$

This matrix is then multiplied by the H vector to give a correction vector Z:

$$Z^{jy}_{b} = \sum_{a} H_a C'^{i}_{ab} \qquad (24)$$

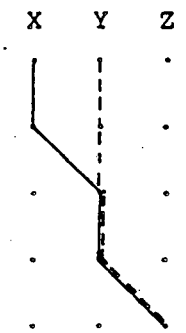
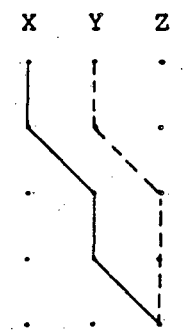
and to account for the symmetric counterpart in the Hamiltonian:

$$Z'^{i}_{ab} = H_a C^{jy}_{b} \qquad (25)$$

The correction subvector Z^{jy} is exactly what is needed, but the second equation yields a matrix Z'^{i} which must be folded to form the true correction subvector Z^{ix} using the inverse operation to that used unfolding the CI vector to form the C' matrix. Thus,

$$Z^{ix}_{ab} = Z'^{i}_{ab} - Z'^{i}_{ba} \qquad a < b \qquad (26)$$

Figure 11. External loop shapes for XY entries to the external space for one-external loops.

Shape	Loop Value		
Iseg =	16	18	22
	A[I(3) + I(1)]	AI(3)	A[I(trl) + BI(tr2)]
	-A[I(3) + I(1)]	-AI(3)	-A[I(trl) + BI(tr2)]

The YX entries are identical except for the interchange of the i and j superscripts. The WY and YW entries are also handled in the same fashion except that the unfolding of the CI vector and the folding of the correction matrix is modified. The CI vector is unfolded to give a symmetric matrix with the diagonal elements multiplied by $\sqrt{2}$. Thus, Equation 23 becomes

$$C'^{i}_{ab} = \begin{cases} C^{iw}_{ab}, & a < b \\ C^{iw}_{ba}, & a > b \\ \sqrt{2}\, C^{iw}_{aa}, & a = b \end{cases} \qquad (27)$$

The folding of the correction matrix Z' is similarly changed:

$$\begin{aligned} Z^{iw}_{ab} &= Z'^{i}_{ab} + Z'^{i}_{ba}, \quad a < b \\ Z^{iw}_{aa} &= \sqrt{2}\, Z'^{i}_{aa} \end{aligned} \qquad (28)$$

This completes all cases involving one-external loops. In the simplest cases, the problem can be expressed as a dot product and a scalar times a vector. In the more complicated cases, the two or three external shapes can be combined into a matrix-vector product

and the outer product of two vectors at the expense of unfolding a section of the CI vector into a matrix and then folding the correction matrix to give the contribution to the correction vector. When the electrons in the external space are triplet coupled (in the configuration with two external electrons), the sign of the loop contribution is removed from the actual value of the contribution and placed instead on the CI elements or the elements of the correction matrix. The sign is taken care of automatically during the unfolding and folding processes, as are factors of $\sqrt{2}$ for the diagonal elements which are singlet coupled.

C. Two-External Loops

The loops with two indices in the external space comprise a rather large fraction of all loops; therefore, it is crucial that they be treated as efficiently as possible. There are seven different ways a two-external loop can enter the external space; of these ways, three actually occur relatively infrequently and are satisfactorily treated by the original shape-driven algorithm. These three entries are XZ, WZ, and YY. The evaluation and processing of the remaining two-external loops, which pass through XX, WX, XW, and WW points at the Fermi level, typically takes 70-80% of the entire computational effort of a calculation. Fortunately, the matrix reformulation handles these loops in about the most efficient fashion possible.

All four entries to the external space involve two configurations with two electrons in virtual orbitals, the four different cases

arising from the spin-couplings of the external orbitals in either configuration. The procedure outlined for the XY, YX, WY, and YW entries for one-external loops can be extended to handle the two-external cases mentioned above. The appropriate section of the CI vector for both the bra and ket walks are unfolding to form matrices of CI coefficients. Again, if the external orbitals are triplet coupled, the matrix of CI coefficients is antisymmetric; if they are singlet coupled, the matrix is symmetric with diagonal elements multiplied by $\sqrt{2}$.

The vector of loop contributions H_a is replaced by a matrix H_{ab} which is formed in a fashion analogous to Equation 22:

$$\begin{aligned} H_{ab} &= A\ I(ijab+tr1) + B\ I(ijab+tr2) \\ H_{ba} &= A\ I(ijab+tr1') + B\ I(ijab+tr2) \end{aligned} \tag{29}$$

for $a < b$. Notice that the offset in the triplet of integrals of the integral multiplying the A coefficient can be different for the upper and lower triangles of this matrix.

Having formed the CI coefficient matrices and the matrix of loop values, correction matrices are formed as follows:

$$\begin{aligned} Z'^{i}_{ab} &= \sum_c H_{ac} C'^{j}_{cb} \\ Z'^{j}_{ab} &= \sum_c H_{ca} C'^{i}_{cb} \end{aligned} \tag{30}$$

Finally, the correction matrices are folded to give the true contri-

bution to the correction vector. As before, the actual method of folding depends on the spin-coupling of the external orbitals in the configurations.

Each matrix product in Equation 30 is the equivalent of the contraction of an $N_e^2 / 2$ portion of the CI vector with an $N_e^2 / 2 \times N_e^2 / 2$ submatrix of the Hamiltonian matrix in a more conventional method. This submatrix is formed of the matrix elements between configurations of the form C_{ab}^i and C_{cd}^j where the internal portions of the configurations differ in two places. Since the matrix elements are identically zero if the two configurations differ in more than four places, only the matrix elements between configurations sharing at least one external orbital in common are nonvanishing. Each configuration interacts with $2N_e$ other configurations: C_{bc}^i can interact with N_e configurations of the form C_{ab}^j and C_{bd}^j as well as with N_e of the C_{ac}^j and C_{cd}^j. Thus, the submatrix contains only N_e^3 nonvanishing matrix elements -- but there are only N_e^2 unique values among the N_e^3 remaining elements. This is because the matrix element between configurations C_{ab}^i and C_{bc}^j does not depend on the orbital c, except for sign and perhaps a normalization constant of $\sqrt{2}$.

The matrix of loop values H_{ab} in Equation 29 is formed from the N_e^2 unique values of loop contributions. That each loop contribution appears N_e times in the true Hamiltonian matrix is reflected in the use of the unique value N_e times in the matrix multiplication of Equation 30. The phase factors and normalization constants of $\sqrt{2}$ are incorporated during the unfolding of the CI vector and the

folding of the correction matrix.

In light of the preceeding discussion, the bulk of the formation of the Hamiltonian matrix can be identified with the formation of the loop value matrix of Equation 29. The matrix products in Equation 30 replaces the usual product of the Hamiltonian matrix with the CI vector. Note that the amount of effort expended evaluating the Hamiltonian matrix scales as N_e^2, while the matrix product scales as N_e^3. Therefore, for large numbers of external orbitals, the time taken evaluating the Hamiltonian matrix is insignificant compared to the actual multiplication by the CI vector.

D. Three-External Loops

These loops are formed by a pair of configurations differing in three places in the external orbitals. Therefore, with the restriction that no more than two external orbitals in any configuration can be occupied, only loops entering the external space at XY and WY points can have three indices in the external space. Furthermore, in the elegant treatment of the one- and two-external loops, many different loops shared identical values. In the three-external loops, there is no such repetition.

However, the form of the loop contribution of the three-external loops is always the same so the work can be done once before the CI calculation begins by forming the following matrices:

$$
\begin{aligned}
P_{ia,bc} &= [ib;ac] + [ic;ab] \\
P_{ia,bb} &= \sqrt{2}\,[ib;ab] \\
P_{ib,ab} &= [ia;bb] + [ib;ab]
\end{aligned}
\tag{31}
$$

and

$$
\begin{aligned}
Q_{ia,bc} &= [ib;ac] - [ic;ab] \\
Q_{ib,ab} &= [ia;bb] - [ib;ab]
\end{aligned}
\tag{32}
$$

There is no restriction on the first pair-index except that i is an internal orbital and an external one. The second pair index is formed from two external orbitals, the first of which always is equal to or above the second on the Shavitt graph.

The P supermatrix is used for evaluating three-external loops entering the external space at WY points; the Q matrix, for those arriving at XY points. The contribution to the correction vector from these loops is given by the following equations:

$$
\begin{aligned}
Z_a^{jy} &= A^i \sum_{bc} P_{ia,bc}\; C_{bc}^{iw} \\
Z_{bc}^{iw} &= \sum_a (A^i\, C_a^{jy})\; P_{ia,bc}
\end{aligned}
\tag{33}
$$

or for XY entries, P is replaced by the Q supermatrix. In the above equations, A^i is the internal portion of the loop coefficient. Notice that since A^i is a constant, it can be removed from the first and premultiplied by the C_a^{jy} coefficients in the second. This again

means that the actual evaluation of Hamiltonian matrix elements is only a small portion of the total work. The internal portion of the loop coefficient appears in Equation 33 $2N_e$ times; of course, forming the supermatrices requires about $N_i N_e^3 /2$ operations for each one, but this process is not repeated every iteration of the Davidson algorithm.

The penalty for this avoided work is the storage of the supermatrices P and Q, each of which is twice as large as the set of integrals it was created from. Except for extremely large basis sets, this additional storage is unlikely to be a significant problem. Also, a certain amount of time will be taken reading the supermatrices into the central memory, etc. For a vector processor, this additional work will be easily recouped by the subsequent vectorization of the three-external loops. The situation is not so clearcut on a purely scalar machine. The benefits of forming the supermatrices may be outweighed by the associated overhead.

E. Four-External Loops

As in the case of the three-external loops, the efficient vectorization of the four-external loops depends on the formation of supermatrices of loop contributions. In this case, however, the loop coefficient is entirely known beforehand and so can be incorporated in the supermatrices. To handle four-external loops where the two configurations are identical in the internal space and pass through the W point at the Fermi level, and thus have singlet coupled electrons in the external space, the following supermatrix is formed:

(34)

$$R_{ab,cd} = [ac;bd] + [ad;bc]$$

$$R_{ac,bc} = [ab;cc] + [ac;bc] + \langle a|h|b\rangle$$

$$R_{ab,cc} = \sqrt{2}\ [ac;bc]$$

$$R_{ab,bc} = [ab;bc] + [ac;bb] + \langle a|h|c\rangle$$

$$R_{ab,ac} = [ab;ac] + [aa;bc] + \langle b|h|c\rangle$$

$$R_{aa,bb} = [ab;ab]$$

$$R_{aa,ab} = \sqrt{2}\ (\ [aa;ab] + \langle a|h|b\rangle\ )$$

$$R_{ab,bb} = \sqrt{2}\ (\ [ab;bb] + \langle a|h|b\rangle\ )$$

The two pair-indices are restricted so that the first of a pair is less than or equal to the second. There is no restriction between the two pair-indices.

The contribution to the correction vector is given by:

$$Z_{ab}^{iw} = \sum_{cd} R_{ab,cd}\, C_{cd}^{iw} \qquad (35)$$

Because there is no restriction between the ab and cd indices, there is no need to exploit the symmetric nature of the Hamiltonian matrix. Four-external loops contribute to submatrices on the main diagonal of the Hamiltonian matrix. Instead of forming just the lower half of the submatrix and then using the symmetric nature of the Hamiltonian to find the contribution from the upper half of the submatrix, it is more convenient to form the entire submatrix.

The corresponding supermatrix for the XX entries is:

$$
\begin{aligned}
S_{ab,cd} &= [ac;bd] - [ad;bc] \\
S_{ac,bc} &= [ab;cc] - [ac;bc] + \langle a|h|b\rangle \\
S_{ab,bc} &= [ab;bc] - [ac;bb] - \langle a|h|c\rangle \\
S_{ab,ac} &= [aa;bc] - [ab;ac] + \langle b|h|c\rangle
\end{aligned}
\tag{36}
$$

The contribution to the correction vector is:

$$
Z_{ab}^{ix} = \sum_{cd} S_{ab,cd}\, C_{cd}^{ix}
$$

The formation of the supermatrices for the four-external loops is advantageous even on a scalar machine. Since the loop coefficients are incorporated in the supermatrices, there is no extra work to be performed each iteration. After the two-external loops, the four-external loops constitute almost all of the rest of the computational effort, especially when using extended basis sets.

This completes the discussion of the matrix reformulation of the shape-driven approach. It should be stressed that this reformulation has nothing to do with the generality of the program, but is rather a convenient way of reorganizing the calculation to reduce the amount of computational work required. Furthermore, the matrix and vector operations are extremely well suited to a vector computer such as the CRAY. A later section will present timings for sample calculations on a CRAY that show the distinct advantage on such a machine of the matrix reformulation.

VII. HIGHER EXCITATIONS

As was mentioned, the preceeding descriptions of both the original and matrix versions of the shape-driven approach were for calculations with configuration lists containing no higher than double excitations from the reference configurations. Yet, it is sometimes desirable to be able to include higher excitations. For example, one may wish to include triply- and quadruply-excited configurations as well. Such a CISDTQ calculation is of interest as a probe of the validity of truncating the CI expansion after double excitations. There are also occasions when, say, a reaction barrier height is controversial, and the inclusion of higher excitations is important for an accurate determination of the barrier height.

The shape-driven approach as outlined can be simply but crudely modified to handle such calculations. Since the external shapes currently programmed handle all cases when both external walks contain two or fewer electrons, the modification is to redefine the external space for each individual loop. The loop-driven search is extended down the loops until they either close or no more than two electrons remain in each walk, at which time, the external space is defined to be the remaining orbitals. Since the supermatrices constructed to handle the three- and four-external loops in the matrix version version depend on the number of external orbitals, the matrix version of the three- and four-external loops cannot readily be used when handling higher excitations. The matrix versions of the one- and two-external loops, however, can be used.

This crude approach to higher excitations results in a substantial degradation of the performance of the program, since a large portion of the evaluation of loops is transferred from the external routines to the relatively inefficient loop-driven code. Furthermore, as the level of excitation increases, the amount of computation done in the loop-driven section increases until, for calculations such as full CI's, the performance can be worse than the previously mentioned slow direct CI program formed by combining the two phases of Brooks' diagonalization tape program into one. The performance can be worse because of the penalty of ignoring upper walks from loops paid in order to be able to page the vectors.

By contrast, the performance should be acceptable when treating low orders of excitations in large orbital spaces. For instance, a calculation containing up through quadruple excitations with perhaps fifty orbitals should be quite reasonable. In principle, calculations with higher excitations could be handled with the same efficiency as singles and doubles by programming the appropriate external shapes. However, the amount of code needed rapidly becomes prohibitive, although it might be possible to include triple and quadruple excitations. The matrix formulation can also be extended to handle at least some of the cases encountered with higher excitations, but it is not obvious that it can be extended to all cases. The matrix version would also need too much code to go beyond about quadruple excitations.

Beyond such quick fixes and brute-force ideas, a major restructuring of the algorithm is required to efficiently handle higher excitations. A promising algorithm would be to recursively search from the middle of loops in both directions. This would be implemented by choosing a search level according to the type of calculation. For instance, for a full CI, the best level would be the middle of the Shavitt graph, while for a multi-reference CISD calculation the optimum level would be the modified Fermi level used in the current programs. The partial walks from the level would be numbered using lexical order for the lower walks and reverse lexical order for the upper walks. The total configuration number of any walk would then be a sum of the lower partial walk's number and an offset determined by the upper partial walk's number.

The searching for, and evaluation of, the loops would always begin at this dividing level rather than at the graph head as is currently done. The program would search upwards compiling a list of all possible upper portions of loops arriving at the search level at a particular pair of vertices. A downward search would then give all the lower portions of loops, each of which would be combined with the list of <u>all</u> possible upper portions to give valid loops. Furthermore, since the upper portion of the graph would be numbered using reverse lexical order and the lower, by lexical order, both upper and lower walks from loops could be exploited without impairing the ability to page the vectors. Naturally, loops entirely within the upper or lower portions of the graph would have to be found by

first searching from the mid-level to the bottom or top of the loop, respectively. This would not amount to much extra work since the number of loops entirely within one portion of the graph is a small percentage of the total and the extra searching required quite minimal.

For a calculation involving no more than double excitations, the current external space routines could be used to evaluate the lower portions of loops. In this limit, the program would function almost identically to the present versions, except that it would be able to exploit the upper walks from loops and should, therefore, execute slightly faster than the present programs. However, as the number of references increases or the level of excitations increases, the suggested algorithm should perform increasingly well compared to the present programs. With a little thought, it would appear that the proposed algorithm could also exploit the repetition of unique values of matrix elements in much the same way that the matrix version does. The difference would be that the matrix version does the repetition through carefully programmed specialized segments of code. The proposed algorithm could realize the same savings by sorting the upper and lower portions of loops. Since this sorting could be implemented in a general fashion, it would handle all cases and would not require the tremendous proliferation of special cases evident in the current method.

VIII. SAMPLE CALCULATIONS

The following sections will present the results of various sample calculations undertaken with the shape-driven approach. First, the discussion will center on a set of calculations on the ethylene molecule. These calculations not only demonstrate the ability of the current programs to handle large CI expansions, but also shed some light on the importance of triply- and quadruply-excited configurations. The next section will be concerned with the comparison of the current programs and the previous programs based on the UGA, as well as the advantage of the matrix reformulation over the original version of the shape-driven approach on a scalar computer. The final section will briefly discuss the preliminary results from the version of the program on the vector CRAY computer.

A. Application To Many-Body Correlation Effects In Ethylene

Recently there has been considerable interest in the importance of many-body correlation effects, mainly because methods capable of handling calculations including such higher-order effects are becoming available. Quantum chemists have hoped that for systems such as closed shell molecules near their equilibrium geometries, the many-body effects are negligible. Otherwise, the prospect of having to include higher excitations is discouraging. The most intensely studied system has been the water molecule, for which the exact solution within a double zeta (DZ) basis set of the electronic Schroedinger equation has been obtained via a full CI calculation.[12f]

The new capabilities of the shape-driven approach have been used in the present study[20] to investigate the importance of triply- and quadruply-excited configurations in ethylene, which serves as the prototype of all unsaturated organic molecules.

Before discussing the calculations, it is necessary to present the geometry and basis set used. The geometry, which is close to the equilibrium structure, assumed the following parameters: $r(C=C) = 1.330\text{Å}$, $r(C-H) = 1.076\text{Å}$, and $\theta(HCH) = 116.6^{\circ}$. This structure results in the following cartesian coordinates (in atomic units) for the atoms: carbons (0.0, 0.0, ±1.25666814) and hydrogens (0.0, ±1.72999314, ±2.32513368).

The basis set used was a standard double zeta plus polarization (DZP) set formed using Dunning's[21] (4s2p) contraction of Huzinaga's[22] (9s4p) basis for carbon and the (2s) contraction of the (4s) basis for hydrogen, the exponents of which were scaled by a factor of 1.2. The polarization functions were a set of six cartesian d functions with orbital exponents of 0.75 on the carbons and a set of three p functions, exponent 1.0 on the hydrogens.

A calculation including all singly- and doubly-excited configurations from the Hartree-Fock reference with the restriction that the two core-like orbitals are always doubly occupied, i.e. frozen, has 5057 configurations. If all triple and quadruple excitations are included, the configuration count soars to 10,593,385 which is clearly too large to be feasible on the Harris-800 minicomputer used for this study. The scheme used to investigate the importance of

the triply- and quadruply-excited configurations was to include only the more important ones. This was accomplished by eliminating from the calculation all configurations with more than two electrons outside the core and valence space. The justification of this scheme is that the valence orbitals are considerably more important than the remaining virtual orbitals for correlating the occupied orbitals. Arguments based on the occupation numbers of the orbitals suggest that this procedure should include 80-90% of the total correlation energy from the triples and quadruples.

The ground state electron configuration of ethylene is the following:

$$1a_g^2\ 1b_{1u}^2\ 2a_g^2\ 2b_{1u}^2\ 1b_{2u}^2\ 1b_{3g}^2\ 3a_g^2\ 1b_{3u}^2$$

where the $1a_g$ and $1b_{1u}$ orbitals are approximately the carbon 1s core orbitals. Beyond these occupied orbitals, the unoccupied orbitals in the valence space are the $1b_{2g}$, $3b_{1u}$, $2b_{2u}$, $4a_g$, $2b_{3g}$, and $4b_{1u}$ orbitals. Since the SCF virtual orbitals do not necessarily have the form desired, a transformation to natural orbitals[23] was employed. The transformation used was that which diagonalized the one-particle density matrix of a CISD calculation with the two core orbitals frozen. This transformation guarantees that the orbitals have the expected form and thus are suitable for the ensuing calculations. Table 4 gives the orbital occupancies for the valence orbitals and the most important virtual orbitals outside the valence space. There is a large gap between the occupied and virtual orbitals in the

valence space and also another significant gap between the valence orbitals and the remaining orbitals, as one would expect.

As mentioned before, three calculations were undertaken using these natural orbitals and always constraining the core orbitals to be doubly occupied. These were a normal CISD from the first natural configuration, and then calculations including through triple excitations and through quadruples. In the last two calculations, all configurations with more than two electrons outside the valence space were eliminated. Another way of viewing the last two calculations would be as multi-reference calculations employing all singles and doubles from all singles, or singles and doubles, in the valence space as references, respectively. The two cases then can be viewed as having 37 and 703 reference configurations. The CISD calculation from the first natural configuration reference had 5057 configurations. In spite of the restriction on the configurations, the triples calculation had 109,473 configurations and the quadruples, 1,046,758. Even though only about 10% of the quadruply-excited configurations were included, the calculation was much larger than any previously attempted.

Table 5 summarizes the results of the calculations, including those of the calculation using the canonical SCF orbitals which was used to find the natural orbitals. Note that there is a slight loss of energy using the natural orbitals in the CISD calculation, but that the difference is quite small. The bulk of the correlation energy is recovered, as expected, from the single reference calculation,

Table 4. Natural orbital occupation numbers (greater than 0.001) for the ground state of ethylene.

Valence Orbitals				Most Important Additional Natural Orbitals	
Occupied in Hartree-Fock Picture		Unoccupied in Hartree-Fock Picture			
$1a_g$	2.000	$1b_{2g}$	0.046	$2b_{3u}$	0.008
$1b_{1u}$	2.000	$3b_{1u}$	0.017	$5a_g$	0.007
$2a_g$	1.982	$2b_{2u}$	0.017	$3b_{2u}$	0.007
$2b_{1u}$	1.976	$4a_g$	0.015	$6a_g$	0.006
$1b_{2u}$	1.973	$2b_{3g}$	0.014	$5b_{1u}$	0.005
$3a_g$	1.969			$1b_{1g}$	0.005
$1b_{3u}$	1.943			$3b_{3u}$	0.004
				$3b_{3g}$	0.003
				$1a_u$	0.003
				$4b_{2u}$	0.002
				$2b_{2g}$	0.002
				$6b_{1u}$	0.002
				$7a_g$	0.002
				$4b_{3g}$	0.001
				$5b_{2u}$	0.001

Table 5. Summary of level of calculation and energies (in hartrees) for the ground state of ethylene at the geometry described in the text and a double zeta plus polarization basis.

	Number of Reference Configurations	Total Number Configurations	Total Energy
Self-Consistent-Field (SCF)	-	1	-78.050 53
Canonical SCF Orbitals, CISD	1	5057	-78.328 11
Above, Davidson Corrected	-	-	-78.354 54*
First Natural Configuration	-	1	-78.049 42
Natural Orbitals, CISD	1	5057	-78.328 00
Above, Davidson Corrected	-	-	-78.354 07*
Triple Excitations	37	109,473	-78.335 01
Quadruple Excitations	703	1,046,758	-78.354 51

* Davidson correction is a nonvariational, approximate correction for the unlinked clusters.

which obtains 0.27858 hartrees of correlation energy. The 37 reference calculation including the more important triple excitations recovers another 0.00700 hartrees, which is only 2.5% as much as the contribution of the single and double excitations. The quadruples add a further 0.01950 hartrees to the total, which is 7.0% of the CISD lowering.

Several interesting points emerge from the above results. The quadruple excitations appear to be almost three times as important as the triples, which is in line with previous results.[12f] Remembering that not all of the triple and quadruple excitations were included, the above figures underestimate somewhat their importance. This, coupled with the neglect of all of the five through sixteen fold excitations, suggests that the CISD calculation recovers probably less than 90% of the total correlation energy. By way of comparison, for the water molecule in a smaller double zeta basis, which lacks the polarization functions of the present study, the singles and doubles account for 94.7% of the correlation energy, the triples and quadruples for 0.77% and 4.4% respectively, leaving only 0.18% for the five through ten fold excitations.[12f] It is reasonable that less of the correlation energy in ethylene is recovered by a simple CISD calculation than in water since ethylene has considerably more electrons. The relative importance of the triples in ethylene compared to the quadruples is considerably different than their importance in water, which is interesting. However, the role of the polarization functions in such calculations is not understood, so it

would be difficult to come to any conculsions at present. A final, and most important question, which has not been addressed in the current study, is whether there are differential effects in the contributions of higher excitations between different structures, or states, of molecules.

These calculations on ethylene were made possible by the ability of the shape-driven programs to page the CI vector. The calculations were performed on the Harris-800 minicomputer, which has sufficient central memory to hold about 190,000 elements of the vectors. Remembering that both the CI and correction vectors must be in core simultaneously, it was necessary to page the vectors for even the 109,473 configuration, 37 reference calculation. For the largest calculation, less than 10% of the vectors could be held. No calculation of this size has been attempted before, and as far as the author is aware, there is no other program in existence which could efficiently handle a calculation with such a small portion of the vectors in core. Test calculations on smaller systems have shown that paging the vectors to the extent used in the largest calculation increases the calculation time by only a few percent.

B. Timing Comparisons With Previuos Programs

Although the shape-driven approach is capable of paging the vectors, for which some price in the form of execution time must be paid, it is nonetheless a fast program. This speed is due to the organization of the processing of loops, which considerably reduces overhead and redundant work. Table 6 compares the time per iteration of the Davidson algorithm for various programs, all of which are based on the unitary group approach. Apart from the SDGUGA program, the programs are a) Brooks' diagonalization tape system,[15c] b) a direct CI program based on the loop-driven approach but exploiting the external space simplifications, and c) a rather slow direct CI program formed by combining the two phases of Brooks' diagonalization tape system into one program. The timings for the diagonalization tape program include an appropriate fraction of the time taken to form the diagonalization tape, since the tape is formed only once and then stored in a disc file.

The first three calculations in Table 6 are relatively small, so the diagonalization tape could be stored. At first thought, one would think that the direct CI programs could not possibly compete favorably with a program which stores the Hamiltonian matrix rather than regenerating it each iteration. However, the SDGUGA program performs about as well as the diagonalization tape program in the first two calculations and considerably better in the third. There are two main reasons for this. The first is that it does take a considerable amount of time to read the diagonalization tape from

Table 6. Timings* for sample calculations when using loop-driven diagonalization tape program, loop-driven direct CI program using external space simplifications, or shape-driven direct CI program.

					Loop-Driven		
Molecule	Sym	Basis	# Orbitals	# Config.	Tape	Direct	SDGUGA
$F + H_2$	C_{2v}	DZP	29	1,125	5.7s	15.5s	6.6s
Al_2	D_{2h}	DZ	22	1,076	11s	25.4s	10.1s
He_4	C_1	5s	20	2,145	40.3s	31.5s	20.6s** 21.1s
H_2CO (TS)	C_s	DZP	38	10,221		300s	188s
$MnCH_2^+$	C_{2v}	TZP	60	22,288	3419s***	855s	601s
C_4H_4	C_{2v}	DZP	56	45,623			1260s
				74,625	(Two references)		1800s
C_2H_4	D_{2h}	DZP	50	5,057		221s	66s
				109,473	(37 references)		40 min.
				1,046,758	(703 references)		13 hr. 10 min.

* Time per iteration of Davidson algorithm, on Harris-800 minicomputer

** Run with program written only for C_1 symmetry

*** Slow direct CI program which generates loops each iteration using loop-driven approach

the disc file, and it is not convenient for the program to have to pick up a matrix element and its indices, and then have to decide what to do with it. The shape-driven approach, on the other hand, implicitly knows which matrix element is being dealt with. The second major drawback with the diagonalization tape program is that it always uses the number of upper and lower walks from a loop. This requires a doubly nested set of DO loops, yet most of the time the number of upper and lower walks are both unity. Thus, the program probably spends more time on the overhead of setting up the DO loops than in the actual computation.

The performance of the loop-driven direct CI program is not as good as the shape-driven program, yet even it manages to outperform the diagonalization tape program on the He_4 calculation. From the relatively poor performance on the D_{2h} Al_2 calculation, it appears that the loop-driven program spends considerable time handling symmetry information. By contrast, symmetry is very readily handled in the SDGUGA program by limiting the ranges of DO loops. The two calculations on He_4 illustrate this. The upper timing is for a program which did not treat symmetry at all; the lower, for the final version, which, for this calculation, must do all the symmetry checking even though there is no symmetry present.

The last four sets of calculations are all larger calculations for which it is impossible to store the diagonalization tape. The SDGUGA program continues to run faster than the loop-driven direct CI program, especially for high symmetry molecules. The one

timing for $MnCH_2^+$ of 3419s is the time taken per iteration of the slow direct CI program formed by combining the two programs which form and then use the diagonalization tape. This result illustrates the importance of using the external space simplifications to speed up the evaluation of Hamiltonian matrix elements.

Although the performance of the original SDGUGA program is impressive, the matrix version can be considerably faster for large calculations, even on a scalar machine. Because the matrix version of the program uses the duplication of matrix elements, it only does about half the computational work that the original program does (for sufficiently large external spaces). The timings presented in Table 7 reflect this reduction in work quite satisfactorily. It appears that for even larger calculations than those presented in the table, the matrix version of the SDGUGA will run in only 30-40% of the time taken by the original program. The extra speed arises from the tremendous simplicity of matrix operations, which means that even on a scalar machine, matrix operations run somewhat faster than other, less structured computations.

Table 7. Timing comparison between the original and the matrix reformulation of the SDGUGA programs. Times given are per iteration of the Davidson algorithm, on the Harris-800 minicomputer.

Molecule	Sym	Basis	# Orbitals	# Configurations	Original	Matrix
N_6	D_{2h}	DZ	48	16,800	276s	200s
N_6	D_{2h}	DZP	84	71,374	29.6 min.	16.7 min.

IX. IMPLEMENTATION OF SDGUGA ON CRAY VECTOR COMPUTER

The CRAY is the first high-performance vector computer to be successful, and as such, it represents a challenge to a computationally bound field like quantum chemistry. Technically, the CRAY is a single-instruction multiple-data machine, i.e. one instruction will cause the same operation to be applied to up to 64 data elements. For very structured processes such as matrix multiplications, one can take advantage of this vector mode to achieve a performance at least an order of magnitude better than any scalar computers can currently manage. It is important for quantum chemistry to be able to harness the power of the vector computers. Yet, many of the algorithms used in quantum chemical calculations do not easily "vectorize". For example, five years after the CRAY was first introduced, the first vectorized CI programs are just starting to work. One of these programs is that of Saunders and van Lenthe,[19] the other is the matrix version of the SDGUGA. None of the other CI programs developed on scalar machines have been successfully vectorized.

The reason for the vectorization of both Saunders' program and the current matrix version of the SDGUGA is the use of matrix and vector operations wherever possible. In a sense, both programs use the same algorithm -- that of folding and unfolding the vectors -- although not enough has been published about Saunders' program to allow a comparison between the two.

It was originally felt that the first version of the shape-driven approach would vectorize, which it indeed did. However, the vector lengths in a typical calculation were so short that only a small increase in speed was noted due to vectorization. Therefore, the matrix version of the program was written explicitly to take advantage of the vector capabilities of the CRAY. The results for a sample calculation were a speed-up in execution time of a factor of fifteen, in part due to the change of algorithm, but mainly due to vectorization. The sample calculation used for test purposes was a CISD calculation on ethylene with a 38 orbital 6-31G* basis and no symmetry. This calculation, which has 29,161 configurations, took 381 seconds on the CRAY for seven iterations using the original program. The matrix version completes the same calculation in 24.4 seconds.

It is quite interesting to examine the breakdown in Table 8 of the time spent doing various phases of the calculation on the CRAY. The first thing to notice is that for the CRAY, this calculation is relatively short, and as a result, a noticeable fraction of the time is spent in overhead starting the calculation. The matrix multiplications which correspond to the contraction of the CI vector with the Hamiltonian matrix account for about half the total time. Yet on the CRAY, the matrix multiplications are more efficiently handled than the other operations, so that on a scalar machine one would expect 85-90% of the computational time to be involved in multiplying the vector by the Hamiltonian matrix. This is the basis of the claim that the time taken to construct matrix elements is negligible in larger calculations.

Table 8. Breakdown of timing for C_2H_4 sample calculation on CRAY. Calculation took 24.4 CPU seconds to do seven iterations.

Calculating diagonal elements (done just once)		3.14%
Loop-driven search for internal portions of loops		7.28%
Extra work due to matrix reformulation		26.68%
Unfolding and folding vectors	20.08%	
Forming matrices of loop contributions	6.60%	
Processing loops		57.01%
Matrix operations	33.45%	
Processing four external loops and part of three external loops	20.04%	
Processing loops not reformulated in matrix form	3.52%	
Overhead		5.89%
		100.00%

One aspect of the program on the CRAY is still not known. Since the computational speed of the CRAY is tremendously faster than conventional computers, but the transfer rate from mass storage to the central memory is no faster, it is not known whether paging the vectors will be practical. It will certainly be tried, and it is probable that improvements can be made in the paging algorithm. If the paging is practical, then calculations with a million configurations will be routine and those with up to about ten million configurations should be possible.

X. CONCLUDING REMARKS

An extremely fast configuration interaction method has been presented which can handle calculations where the CI and correction vectors cannot simultaneously be held in central memory. The method derives its speed from the efficient use of the simplification of the external space for calculations involving only single and double excitations from a set of reference configurations. The ability to page the vectors is due to the use of lexical order and from ignoring the number of upper walks from loops. The latter factor increases the number of loops which must be evaluated, but this increase is more than compensated for by the speed of the shape-driven approach. A second version of the program has been formulated almost entirely in terms of small vector and matrix operations, and is therefore extremely well suited to a vector computer or an array processor.

In the matrix reformulation, it is possible to exploit the fact that unique-valued matrix elements are repeated many times in the Hamiltonian matrix. By forming these matrix elements only once and then using them many times, it is possible for the construction of the Hamiltonian matrix to become a rather small portion of the entire calculation. In the limit of a large basis, all of the computational effort is consumed by the multiplication of the elements of the CI vector by the Hamiltonian matrix elements. It would appear that if this is indeed the situation, then the shape-driven approach yields a program which is essentially as fast as any CI program can be if it uses the entire Hamiltonian matrix.

In the last sections, the timings and results of various sample calculations have been presented. The first set of these calculations demonstrate the ability of the programs to page the vectors. The calculations suggest that for a prototype unsatruated organic molecule, ethylene, a CI including all single and double replacements from the Hartree-Fock reference recovers just under 90% of the correlation energy, while triples and quadruples account for about 2% and 6% of the correlation energy. This confirms the commonly held belief that triples and quadruples are not very important, but does not rule out the possibility of a noticeable differential effect between different geometries or states.

Further examples demonstrate that for relatively small CI expansions, the SDGUGA program is faster than previous unitary group based programs, which in turn, are faster than more conventional programs. Furthermore, the matrix reformulation is shown to be about twice as fast as the original program for larger calculations. Preliminary results of the implementation on the vector CRAY computer are very encouraging, with a factor of 15 being gained from vectorization.

The aim of this work has been the development of an algorithm for CI calculations which can use the computing power available today to handle large CI calculations. There is a large demand for accurate calculations on chemical systems as well as a need to understand the effects of many-body correlations. The current program goes a long way toward solving the first problem. Until now, there has been a

considerable amount of art involved with large-scale CI calculations. This can be contrasted with SCF calculations, which have become routine and the SCF programs, which have almost become "black-boxes". It is hoped that the current programs will be the beginning of a trend towards "black-box" CI programs which can be easily distributed and used. The only limitation on a calculation should be the cost, i.e. the amount of time required. The amount of peripheral storage and central memory available should not be limitations. The current programs meet these requirements since there is no diagonalization tape to store and the vectors can be paged.

On the second aspect, higher excitations, the current programs are much more limited. Although they can handle higher excitations, the current programs lose their efficiency for such calculations. Various methods have been proposed for overcoming these defects, the most promising of which involves a bidirectional search for loops, beginning at the middle of the graph. The experience gained with paging the vectors in the current programs will be invaluable, since calculations with higher excitations tend to have extremely long configuration lists.

The SDGUGA programs are capable of handling almost any calculation now desired. By running large calculations, defects in the current programs may become obvious and improvements, or indeed completely new algorithms, may be developed. Until such calculations are tried, no one really knows what their requirements will be. This

will perhaps be the largest contribution of the current work: to convince quantum chemists that large CI calculations are possible, and to continue the learning process needed to develop yet better methods.

Appendix 1: The Loop-Searching Master Table

The following pages contain a list of all possible segment shapes, the segment-shape values, and other information useful in the evaluation of loops. Each entry in the table is described below.

ISEG — Divides the table into twenty-two major sections, each of which contains related segment shapes. The first section is comprised of the sixteen different ways to open a loop; the subsequent sections gather together segments which are valid continuations of a particular shape of partial loop.

JSEG — A simple index numbering the 228 segment shapes.

NEXT — Indicates the ISEG value of valid segments for the continuation of the partial loop terminated by the current segment. A value of zero indicates that the current segment closes the loop.

SS — The case values of the bra and ket sides of the current segment.

TRACK — Information used to determine the form of the loop contribution. The TRACK value is only occasionally modified during the evaluation of a loop. The last value encountered determines the form of the loop contribution. A dash indicates that the TRACK value is not changed by the segment being processed. The TRACK value is either a single number,

a pair of numbers separated by a comma, or two or three numbers enclosed in parentheses. The loop contribution, v, has the following form for the three cases:

```
      n:  v = XI(ijkl + n)
    n,m:  v = X[I(ijkl + n) + ZI(ijkl + m)]
  (n,m):  v = X[I(ijkl + n) + I(ijkl + m)]
```

X and Z are the loop coefficients (see below) and I(ijkl+n) is the integral with an offset of "n" in the group of integrals at address ijkl.

JKCOND These two entities (JCOND and KCOND) determine when the j and k indices of the loop are reached. The i-level is always the level of the loop opening and the l-level, that of closing, but the j- and k-levels are determined by the shape of the loop. JCOND has the following values:

```
-1  if the j-level has not yet been reached
 1  if the current level is the j-level
 0  if the j-level has been passed already.
```

KCOND has only two possible values:

```
 1  if the current level is the k-level
 0  if the current level is not the k-level.
```

The loop indices i, j, k, and l are used in the evaluation of the integral group address ijkl.

CODE — The CODE value is related to the shape value listed under the heading VALUE. The programs use the CODE value to determine which section of program to use in evaluating the segment-shape values.

VALUE — These algebraic expressions give the partial product of the segment-shape values in terms of the previous product and a set of auxiliary functions defined below. In the body of this work only two values, A and B, were mentioned; however, for convenience, the current table employs three coefficients. These three are X, Y, and Z = Y/X. All loops involve the X coefficient, but only those two-electron loops which involve a summation over triplet and singlet recoupled states use the last two coefficients. Over the range of overlap, there are two coefficients, but outside of the range of overlap, the ratio of these two is constant. Therefore, it is convenient to remember the ratio and keep track of only the X coefficient. Z is the ratio, and is set only once during the evaluation of a loop.

The shorthand used in the segment values consists of the following constant and functions:

$$t = \sqrt{\tfrac{1}{2}}$$

$$A(p,q) = \sqrt{\frac{b+p}{b+q}}$$

$$B(p) = \sqrt{\frac{2}{(b+p)\quad(b+p+1)}}$$

$$C(p) = \frac{\sqrt{(b+p-1)(b+p+1)}}{b+p}$$

$$D(p) = \sqrt{\frac{(b+p-1)(b+p+2)}{(b+p)(b+p+1)}}$$

$$E(p) = \sqrt{\frac{2}{(b+p)(b+p+1)}}$$

The parameter b is the b-value of the vertex of the Shavitt graph that the ket walk passes through at the top of the segment.

Both the LDGUGA and SDGUGA CI programs require that the integrals be sorted into the following seven groups:

Type	Offset	Integral	Indices
1	1	[ik;jl]	i > j > k > l
	2	[ij;kl]	
	3	[il;jk]	
2	1	[ij;jl]	i > j = k > l
	2	[il;jj]	
	3	not used	
3	1	[ik;il]	i = j > k > l
	2	[ii;kl]	
4	1	[il;jl]	i > j > k = l
	2	[ij;ll]	
	3	not used	
5	1	[ii;il]	i = j = k > l
	2	[il;ll]	i > j = k = l
	3	$\langle i\|h\|l\rangle$	
6	1	[il;il]	i = j > k = l
	2	[ii;ll]	
7	1	[ii;ii]	i = j = k = l
	2	$\langle i\|h\|i\rangle$	

The address of any particular integral is found by first sorting the indices into strictly descending order (except for integrals such as [il;ll], which is stored under the indices iiil). If the ordered indices are i, j, k, and l and the corresponding symmetries of the orbitals are Γ_i, etc., then the address of the group of integrals is:

$$ijkl = IJADD[\tfrac{1}{2}i(i-1) + j] + KADD(k, \Gamma_i \times \Gamma_j) + LADD(l, \Gamma_i \times \Gamma_j \times \Gamma_k)$$

The offset of the particular integral is then determined from the table of integral groups and is added to the group address.

This integral storage scheme is particularly useful when evaluating loops since the loop indices are the sorted indices needed to evaluate the integral group address. The TRACK value then determines the form of the loop contribution expression and the offsets of the integrals within the integral group.

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
1	1	3	01	1	-10	1	X' = 1
	2	18	01	3	11	1	X' = 1
	3	2	02	1	-10	1	X' = 1
	4	17	02	3	11	1	X' = 1
	5	10	03	1	10	1	X' = 1
	6	0	11	2	11	1	X' = 1
	7	5	11	1,2	10	44	$\begin{cases} X' = t \\ Y' = -tA(-1,1) \end{cases}$
	8	7	12	1	10	1	X' = 1
	9	2	13	1	-10	3	X' = A(0,1)
	10	15	13	(3,1)	11	3	X' = A(0,1)
	11	0	22	2	11	1	X' = 1
	12	5	22	1,2	10	45	$\begin{cases} X' = t \\ Y' = tA(3,1) \end{cases}$
	13	3	23	1	-10	4	X' = A(2,1)
	14	16	23	(3,1)	11	4	X' = A(2,1)
	15	0	33	2,1	11	50	$\begin{cases} X' = 2 \\ Z' = \frac{1}{2} \end{cases}$
	16	4	33	1,2	10	51	$X' = \sqrt{2}$

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
2	17	2	00	-	-10	1	X' = X
	18	11	01	1,3	10	40	X' = XtA(1,0) Y' = -XtA(-1,0)
	19	12	02	1,3	10	1	X' = X
	20	7	10	3	10	1	X' = X
	21	2	11	-	-10	6	X' = XC(0)
	22	21	11	2	11	6	X' = XC(0)
	23	12	13	1,3	10	7	X' = XA(-1,0)
	24	6	20	1,2	10	46	X' = Xt Y' = XtA(2,0)
	25	3	21	-	-10	9	X' = X/b
	26	22	21	2,1	11	54	X' = X/b Z' = b
	27	2	22	-	-10	2	X' = -X
	28	21	22	2,1	11	52	X' = -X Z' = -1
	29	11	23	1,3	10	41	X' = -Xt Y' = XtA(2,0)
	30	20	30	-	11	5	X' = XA(1,0)
	31	6	31	1,2	10	47	X' = XtA(1,0) Y' = XtA(-1,0)
	32	7	32	3	10	8	X' = -XA(1,0)
	33	2	33	-	-10	2	X' = -X
	34	21	33	2,1	11	53	X' = -2X Z' = -½

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
3	35	3	00	-	-10	1	X' = X
	36	13	01	1,3	10	1	X' = X
	37	11	02	1,3	10	42	X' = XtA(1,2) Y' = XtA(3,2)
	38	6	10	1,2	10	48	X' = Xt Y' = -XtA(0,2)
	39	3	11	-	-10	2	X' = -X
	40	22	11	2,1	11	52	X' = -X Z' = -1
	41	2	12	-	-10	36	X' = -X/(b+2)
	42	21	12	2,1	11	55	X' = -X/(b+2) Z' = -(b+2)
	43	11	13	1,3	10	43	X' = -Xt Y' = -XtA(0,2)
	44	9	20	3	10	1	X' = X
	45	3	22	-	-10	11	X' = XC(2)
	46	22	22	2,1	11	11	X' = XC(2)
	47	13	23	1,3	10	12	X' = XA(3,2)
	48	19	30	-	11	10	X' = XA(1,2)
	49	9	31	3	10	13	X' = -XA(1,2)
	50	6	32	1,2	10	49	X' = XtA(1,2) Y' = -XtA(3,2)
	51	3	33	-	-10	2	X' = -X
	52	22	33	2,1	11	53	X' = -2X Z' = -½

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
4	53	4	00	-	00	1	$X' = X$
	54	22	01	-	01	77	$\begin{cases} X' = -Xt \\ Z' = -2 \end{cases}$
	55	21	02	-	01	77	$\begin{cases} X' = -Xt \\ Z' = -2 \end{cases}$
	56	0	11	-	01	77	$\begin{cases} X' = -Xt \\ Z' = -2 \end{cases}$
	57	4	11	-	00	1	$X' = X$
	58	21	13	-	01	79	$\begin{cases} X' = -XtA(0,1) \\ Z' = -2 \end{cases}$
	59	0	22	-	01	77	$\begin{cases} X' = -Xt \\ Z' = -2 \end{cases}$
	60	4	22	-	00	1	$X' = X$
	61	22	23	-	01	80	$\begin{cases} X' = -XtA(2,1) \\ Z' = -2 \end{cases}$
	62	0	33	-	01	78	$\begin{cases} X' = -\sqrt{2}\ X \\ Z' = -2 \end{cases}$
	63	4	33	-	00	1	$X' = X$

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
5	64	5	00	-	00	71	$\begin{cases} X' = X \\ Y' = Y \end{cases}$
	65	22	01	-	01	67	$\begin{cases} X' = -Xt + YtA(2,0) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	66	21	02	-	01	68	$\begin{cases} X' = -Xt - YtA(0,2) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	67	0	11	1	01	87	$\begin{cases} X' = -Xt + YtA(2,0) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	68	5	11	-	00	75	$\begin{cases} X' = X \\ Y' = YD(0) \end{cases}$
	69	7	12	-	00	83	$X' = -YE(0)$
	70	21	13	-	01	69	$\begin{cases} X' = -XtA(0,1) - YtA(2,1) * \\ Z' = \sqrt{2}XA(0,1)/X' \end{cases}$
	71	0	22	-	01	68	$\begin{cases} X' = -Xt - YtA(0,2) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	72	5	22	-	00	76	$\begin{cases} X' = X \\ Y' = YD(1) \end{cases}$
	73	22	23	-	01	70	$\begin{cases} X' = -XtA(2,1) + YtA(0,1) * \\ Z' = \sqrt{2}XA(2,1)/X' \end{cases}$
	74	0	33	-	01	82	$\begin{cases} X' = -\sqrt{2}X \\ Z' = -2 \end{cases}$
	75	5	33	-	00	71	$\begin{cases} X' = X \\ Y' = Y \end{cases}$

* If X'=0, change TRACK to 2 and set X'=Z'X'

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
6	76	6	00	-	00	71	$\begin{cases} X' = X \\ Y' = Y \end{cases}$
	77	22	01	3,2	01	67	$\begin{cases} X' = -Xt + YtA(2,0) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	78	21	02	3,2	01	68	$\begin{cases} X' = -Xt - YtA(0,2) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	79	20	10	-	01	67	$\begin{cases} X' = -Xt + YtA(2,0) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	80	0	11	-	01	87	$\begin{cases} X' = -Xt + YtA(2,0) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	81	6	11	-	00	75	$\begin{cases} X' = X \\ Y' = YD(0) \end{cases}$
	82	7	12	3	00	83	$X' = -YE(0)$
	83	21	13	3,2	01	69	$\begin{cases} X' = -XtA(0,1) - YtA(2,1) * \\ Z' = \sqrt{2XA(0,1)}/X' \end{cases}$
	84	19	20	-	01	68	$\begin{cases} X' = -Xt - YtA(0,2) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	85	9	21	3	00	83	$X' = -YE(0)$
	86	0	22	-	01	68	$\begin{cases} X' = -Xt - YtA(0,2) * \\ Z' = \sqrt{2}X/X' \end{cases}$
	87	6	22	-	00	76	$\begin{cases} X' = X \\ Y' = YD(0) \end{cases}$
	88	22	23	3,2	01	70	$\begin{cases} X' = -XtA(2,1) + YtA(0,1) * \\ Z' = \sqrt{2XA(2,1)}/X' \end{cases}$
	89	19	31	-	01	69	$\begin{cases} X' = -XtA(0,1) - YtA(2,1) * \\ Z' = \sqrt{2XA(0,1)}/X' \end{cases}$
	90	20	32	-	01	70	$\begin{cases} X' = -XtA(2,1) + YtA(0,1) * \\ Z' = \sqrt{2XA(2,1)}/X' \end{cases}$
	91	0	33	-	01	82	$\begin{cases} X' = -\sqrt{2}X \\ Z' = -2 \end{cases}$
	92	6	33	-	00	71	$\begin{cases} X' = X \\ Y' = Y \end{cases}$

* If X'=0, change TRACK to 2 and set X'=Z'X'

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
7	93	7	00	-	00	1	X' = X
	94	21	01	-	01	6	X' = XC(0)
	95	7	11	-	00	16	X' = -XC(0)
	96	20	20	1	01	6	X' = XC(0)
	97	0	21	1	01	6	X' = XC(0)
	98	8	21	-	00	17	X' = $-\sqrt{2}X/b$
	99	7	22	-	00	16	X' = -XC(0)
	100	21	23	-	01	74	X' = -XA(-1,0)
	101	20	31	1	01	8	X' = -XA(1,0)
	102	7	33	-	00	1	X' = X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
8	103	8	00	-	00	1	X' = X
	104	22	01	-	01	18	X' = XtA(2,0)
	105	21	02	-	01	19	X' = -XtA(0,2)
	106	20	10	1	01	18	X' = XtA(2,0)
	107	0	11	1	01	18	X' = XtA(2,0)
	108	8	11	-	00	22	X' = XD(0)
	109	7	12	-	00	24	X' = -XE(0)
	110	21	13	-	01	20	X' = -XtA(2,1)
	111	19	20	1	01	19	X' = -XtA(0,2)
	112	9	21	-	00	24	X' = -XE(0)
	113	0	22	1	01	19	X' = -XtA(0,2)
	114	8	22	-	00	23	X' = XD(1)
	115	22	23	-	01	21	X' = XtA(0,1)
	116	19	31	1	01	20	X' = -XtA(2,1)
	117	20	32	1	01	21	X' = XtA(0,1)
	118	8	33	-	00	1	X' = X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
9	119	9	00	-	00	1	X' = X
	120	22	02	-	01	11	X' = XC(2)
	121	19	10	1	01	11	X' = XC(2)
	122	9	11	-	00	27	X' = -XC(2)
	123	0	12	1	01	11	X' = XC(2)
	124	8	12	-	00	28	X' = $-\sqrt{2}$X/(b+2)
	125	22	13	-	01	81	X' = -XA(3,2)
	126	9	22	-	00	27	X' = -XC(2)
	127	19	32	1	01	13	X' = -XA(1,2)
	128	9	33	-	00	1	X' = X
10	129	10	00	-	00	1	X' = X
	130	21	10	-	01	3	X' = XA(0,1)
	131	10	11	-	00	2	X' = -X
	132	22	20	-	01	4	X' = XA(2,1)
	133	10	22	-	00	2	X' = -X
	134	0	30	-	01	1	X' = X
	135	22	31	-	01	2	X' = -X
	136	21	32	-	01	2	X' = -X
	137	10	33	-	00	1	X' = X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
11	138	11	00	-	00	71	$\begin{cases} X' = X \\ Y' = Y \end{cases}$
	139	21	10	-	00	63	$\begin{cases} X' = XtA(0,1) + YtA(2,1) \\ Z' = (-XtA(0,1) - YtA(2,1))/X' \end{cases}$ *
	140	11	11	-	00	72	$\begin{cases} X' = -X \\ Y' = -YD(0) \end{cases}$
	141	12	12	-	00	84	$X' = YB(1)$
	142	22	20	-	01	65	$\begin{cases} X' = XtA(2,1) - YtA(0,1) \\ Z' = (XtA(2,1) + YtA(0,1))/X' \end{cases}$ *
	143	13	21	-	00	85	$X' = YB(0)$
	144	11	22	-	00	73	$\begin{cases} X' = -X \\ Y' = -YD(1) \end{cases}$
	145	0	30	1	01	29	$X' = 2X$
	146	22	31	-	01	66	$\begin{cases} X' = -Xt + YtA(2,0) \\ Z' = (-Xt - YtA(2,0))/X' \end{cases}$ *
	147	21	32	-	01	64	$\begin{cases} X' = -Xt - YtA(0,2) \\ Z' = (-Xt + YtA(0,2))/X' \end{cases}$ *
	148	11	33	-	00	71	$\begin{cases} X' = X \\ Y' = Y \end{cases}$

* If X'=0, set TRACK to 3 and set X'=Z'X'

12	149	12	00	-	00	1	$X' = X$
	150	12	11	-	00	30	$X' = XD(-1)$
	151	21	20	-	01	56	$\begin{cases} X' = X \\ Z' = -1 \end{cases}$
	152	14	21	-	00	86	$X' = XB(-1)$
	153	12	22	-	00	1	$X' = X$
	154	21	31	-	01	57	$\begin{cases} X' = XA(1,0) \\ Z' = -1 \end{cases}$
	155	12	33	-	00	1	$X' = X$

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
13	156	13	00	-	00	1	X' = X
	157	22	10	-	01	56	X' = X Z' = -1
	158	13	11	-	00	1	X' = X
	159	14	12	-	00	32	X' = XB(2)
	160	13	22	-	00	31	X' = XD(2)
	161	22	32	-	01	58	X' = XA(1,2) Z' = -1
	162	13	33	-	00	1	X' = X
14	163	14	00	-	00	1	X' = X
	164	21	10	-	01	59	X' = XtA(2,1) Z' = -1
	165	14	11	-	00	33	X' = -XD(0)
	166	12	12	-	00	34	X' = XB(1)
	167	22	20	-	01	61	X' = -XtA(0,1) Z' = -1
	168	13	21	-	00	35	X' = XB(0)
	169	14	22	-	00	88	X' = -XD(1)
	170	22	31	-	01	62	X' = XtA(2,0) Z' = -1
	171	21	32	-	01	60	X' = -XtA(0,2) Z' = -1
	172	14	33	-	00	1	X' = X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
15	173	15	00	-	00	1	X' = X
	174	15	11	-	00	6	X' = XC(0)
	175	0	20	-	00	1	X' = X
	176	16	21	-	00	9	X' = X/b
	177	15	22	-	00	2	X' = -X
	178	0	31	(3,2,1)	00	5	X' = XA(1,0)
	179	15	33	-	00	2	X' = -X
16	180	16	00	-	00	1	X' = X
	181	0	10	-	00	1	X' = X
	182	16	11	-	00	2	X' = -X
	183	15	12	-	00	36	X' = -X/(b+2)
	184	16	22	-	00	11	X' = XC(2)
	185	0	32	(3,2,1)	00	10	X' = XA(1,2)
	186	16	33	-	00	2	X' = -X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
17	187	17	00	-	00	1	X' = X
	188	17	11	-	00	6	X' = XC(0)
	189	0	20	-	00	1	X' = X
	190	18	21	-	00	9	X' = X/b
	191	17	22	-	00	2	X' = -X
	192	0	31	(3,2)	00	5	X' = XA(1,0)
	193	17	33	-	00	2	X' = -X
18	194	18	00	-	00	1	X' = X
	195	0	10	-	00	1	X' = X
	196	18	11	-	00	2	X' = -X
	197	17	12	-	00	36	X' = -X/(b+2)
	198	18	22	-	00	11	X' = XC(2)
	199	0	32	(3,2)	00	10	X' = XA(1,2)
	200	18	33	-	00	2	X' = -X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
19	201	19	00	-	00	1	X' = X
	202	0	02	-	00	1	X' = X
	203	19	11	-	00	37	X' = XC(1)
	204	20	12	-	00	38	X' = X/(b+1)
	205	0	13	-	00	4	X' = XA(2,1)
	206	19	22	-	00	2	X' = -X
	207	19	33	-	00	2	X' = -X
20	208	20	00	-	00	1	X' = X
	209	0	01	-	00	1	X' = X
	210	20	11	-	00	2	X' = -X
	211	19	21	-	00	39	X' = -X/(b+2)
	212	20	22	-	00	37	X' = XC(1)
	213	0	23	-	00	3	X' = XA(0,1)
	214	20	33	-	00	2	X' = -X

ISEG	JSEG	NEXT	SS	TRACK	JKCOND	CODE	VALUE
21	215	21	00	-	00	1	X' = X
	216	21	11	-	00	6	X' = XC(0)
	217	0	20	-	00	1	X' = X
	218	22	21	-	00	9	X' = X/b
	219	21	22	-	00	2	X' = -X
	220	0	31	-	00	5	X' = XA(1,0)
	221	21	33	-	00	2	X' = -X
22	222	22	00	-	00	1	X' = X
	223	0	10	-	00	1	X' = X
	224	22	11	-	00	2	X' = -X
	225	21	12	-	00	36	X' = -X/(b+2)
	226	22	22	-	00	11	X' = XC(2)
	227	0	32	-	00	10	X' = XA(1,2)
	228	22	33	-	00	2	X' = -X

Appendix 2: The External Shapes

This appendix contains the diagrams and partial loop coefficients for all possible external portions of loops. It does not contain the four-external loops. The external shapes are grouped according to the vertices passed through at the Fermi level and by the ISEG value from the internal portion of the loop. Please note that the loop coefficients are given different symbols here than in the loop-searching master table in Appendix 1. Thus X becomes A, and Z is B. Y remains Y, although it is seldom used. Also, in this section, the variables A, B, and Y refer to the value from the internal portion of the loop. The contribution from the external shape is explicitly included in the formula for the loop contributions.

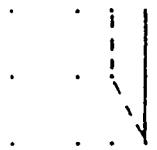
Beside each diagram are the indices of the loop which are in the external space, as well as an indication of the total loop indices. Thus, a notation "I=ijab" means that the loop indices are i and j in the internal space, and a and b in the external space. These indices also define the integral group address as was detailed in Appendix 1. The form of the loop contribution is listed beside the diagrams under the appropraite ISEG value. I(n) is the integral in the integral group defined by the loop indices with an offset of n.

YZ Entries, ISEG= 16 18 22

A[I(3) + I(1)] AI(3) A[I(tr1) + BI(tr2)]

I=ijka

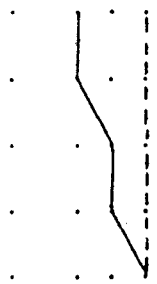
ZY Entries, ISEG= 20



A[I(tr1) + BI(tr2)]

I=ijka

XZ Entries, ISEG= 13



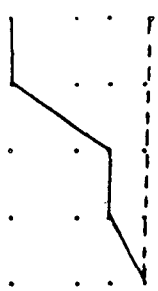
A[I(1) - I(3)]

I=ijab

WZ Entries, ISEG= 10 11

AI(1) $\sqrt{2}$ AI(1)

I=ijaa



$\sqrt{2}$ AI(1) A[I(1) + I(3)]

I=ijab

XY Entries. [SEG= 3

-A[I(1) - I(2)]

I=iaab

A[I(1) - I(2)]

I=iabb

A[I(1) - I(2)]

I=iabc

A[I(3) - I(2)]

I=iabc

XY Entries, ISEG= 3 , continued.

A[I(3) - I(1)]

I=iabc

ISEG= 16 18 22

-A[I(3) + I(1)] -AI(3) -A[I(tr1) + BI(tr2)]

I=ijkb

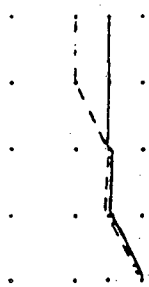
A[I(3) + I(1)] AI(3) A[I(tr1) + BI(tr2)]

I=ijka

YX Entries, ISEG= 20

a -A[I(tr1) + BI(tr2)]

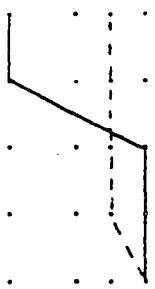
b I=ijkb



a A[I(tr1) + BI(tr2)]

b I=ijka

WY Entries, ISEG= 2



a $\sqrt{2}$ AI(1)

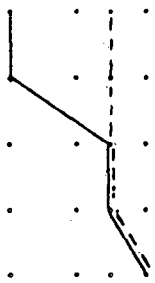
b I=iaab

a $\sqrt{2}$ AI(1)

b I=iabb

a A[I(1) + I(2)]

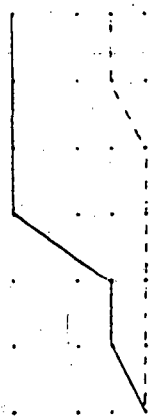
b I=iaab



a A[I(1) + I(2)]

b I=iabb

WY Entries, ISEG= 2 , continued.



A[I(1) + I(3)]

I=iabc

A[I(1) + I(2)]

I=iabc

A[I(3) + I(2)]

I=iabc

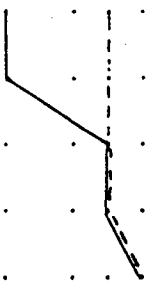
WY Entries, (continued) ISEG= 15 17 21

a I=ijka $\sqrt{2}$ A[I(1) + I(2) +I(3)] $\sqrt{2}$ A[I(tr1) + BI(tr2)]

$\sqrt{2}$ A[I(3) + I(2)]

a A[I(3) + I(1)] AI(3) A[I(tr1) + BI(tr2)]

b I=ijkb



a A[I(3) + I(1)] AI(3) A[I(tr1) + BI(tr2)]

b I=ijka

YW Entries, ISEG= 19

$\sqrt{2}\,A[I(tr1) + BI(tr2)]$

a I=ijka

a $A[I(tr1) + BI(tr2)]$

b I=ijkb

a $A[I(tr1) + BI(tr2)]$

b I=ijka

WX Entries, ISEG= 7

$-\sqrt{3/2}$ AI(1)

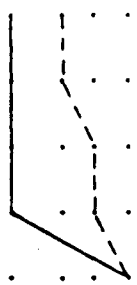
I=ijab

$(\sqrt{3}/2)$AI(1)

I=ijab

$(\sqrt{3}/2)$AI(trl)

I=ijab



$\sqrt{3/2}$ AI(trl)

I=ijab

WX Entries, ISEG= 7 , continued.

$(\sqrt{3}\ /2)$AI(1)

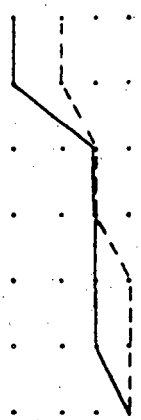
I=ijaa

$-(\sqrt{3}\ /2)$AI(1)

I=ijbb

$(\sqrt{3}\ /2)$AI(trl)

I=ijac



$-(\sqrt{3}\ /2)$AI(trl)

I=ijbc

WX Entries, ISEG= 7 , continued.

$-(\sqrt{3}/2)AI(1)$

I=ijbc

$-(\sqrt{3}/2)AI(1)$

I=ijac

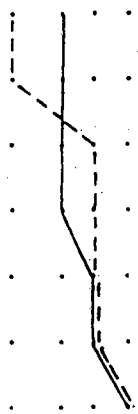
XW Entries, ISEG= 9

$-\sqrt{3/2}$ AI(tr1)

I=ijab

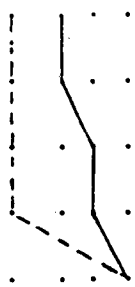
$(\sqrt{3}/2)$ AI(1)

I=ijab



$(\sqrt{3}/2)$ AI(tr1)

I=ijab



$\sqrt{3/2}$ AI(1)

I=ijab

XW Entries, ISEG= 9 , continued.

a $(\sqrt{3}/2)$ AI(1)

b I=ijaa

a $-(\sqrt{3}/2)$ AI(1)

b I=ijbb

a

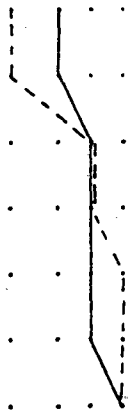
b $(\sqrt{3}/2)$ AI(1)

c I=ijac

XW Entries, ISEG= 9 , continued.

$-(\sqrt{3}/2)$ AI(1)

I=ijbc



$-(\sqrt{3}/2)$ AI(trl)

I=ijbc

$-(\sqrt{3}/2)$ AI(trl)

I=ijac

YY Entries, ISEG = 4 5

Define: $C = -\sqrt{\frac{1}{2}}\,A + \sqrt{\frac{3}{2}}\,Y$

$-\sqrt{\frac{1}{2}}\,A[I(1) - 2I(2)]$ $\quad CI(1) + 2AI(2)$

I=ijab

$-\sqrt{\frac{1}{2}}\,A[I(1) - 2I(2)]$ $\quad CI(1) + 2AI(2)$

I=ijaa

YY Entries, ISEG= 6 8

(continued) Define: $C = -\sqrt{(1/2)}\ A + \sqrt{(3/2)}\ Y$

$CI(1) + \sqrt{2}\ AI(2)$ $\sqrt{\frac{3}{2}}AI(1)$

I=ijab

$CI(3) + \sqrt{2}\ AI(2)$ $\sqrt{\frac{3}{2}}AI(trl)$

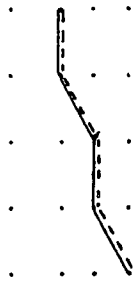
I=ijab

$CI(1) + \sqrt{2}\ AI(2)$ $\sqrt{\frac{3}{2}}AI(1)$

I=ijaa

XX Entries, ISEG= 4 5

Define: $C = -\sqrt{\frac{1}{2}}\,A + Y$



a $\quad -\sqrt{\frac{1}{2}}\,A[I(1) - 2I(2)] \qquad CI(1) + \sqrt{2}\,AI(2)$

b $\quad I=ijaa$

a $\quad -\sqrt{\frac{1}{2}}\,A[I(1) - 2I(2)] \qquad CI(1) + \sqrt{2}\,AI(2)$

b $\quad I=ijbb$

a $\quad \sqrt{\frac{1}{2}}\,A[I(1) - 2I(2)] \qquad -CI(1) - \sqrt{2}\,AI(2)$

b

c $\quad I=ijac$

XX Entries, ISEG= 4 5 , continued.

Define: $C = -\sqrt{\frac{1}{2}}\,A + Y$

a $\quad -\sqrt{\frac{1}{2}}\,A[I(1) - 2I(2)] \qquad CI(1) + \sqrt{2}\,AI(2)$

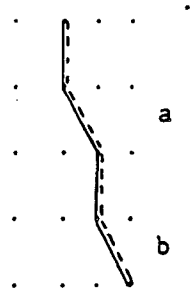
b

c $\quad I=ijab$

XX Entries, ISEG= 6 8

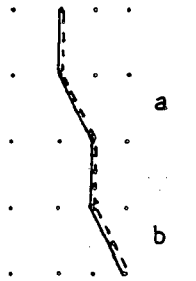
(continued) Define: $C = -\sqrt{\frac{1}{2}}\,A + Y$

$D = \sqrt{2}\,A$



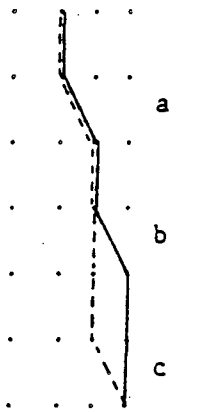
a CI(1) + DI(2) AI(1)

b I=ijaa



a CI(1) + DI(2) AI(1)

b I=ijbb



a

b CI(1) + DI(2) AI(1)

c I=ijbc

XX Entries, ISEG= 6 8, continued.

I=ijbc CI(3) + DI(2) AI(trl)

I=ijac -CI(1) - DI(2) -AI(1)

I=ijac -CI(3) - DI(2) -AI(trl)

XX Entries, ISEG= 6 8 , continued.

CI(3) + DI(2) AI(trl)

I=ijab

CI(1) + DI(2) AI(1)

I=ijab

WW Entries, ISEG= 4 & 5 6

Define: $E = -\sqrt{\frac{1}{2}}\,A$

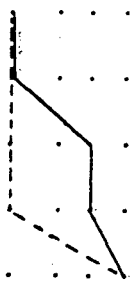
a I=ijaa $2E[I(1) - 2I(2)]$ $2E[I(1) - 2I(2)]$

a --- $\sqrt{2}\,E[I(1) - 2I(2)]$

b i=ijab

a $\sqrt{2}\,E[I(1) - 2I(2)]$ $\sqrt{2}\,E[I(3) - 2I(2)]$

b I=ijab



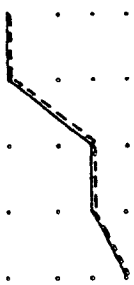
a --- $\sqrt{2}\,E[I(1) - 2I(2)]$

b I=ijab

WW Entries, ISEG= 4 & 5 6, continued.

a $\sqrt{2}\ E[I(1) - 2I(2)]$ $\sqrt{2}\ E[I(3) - 2I(2)]$

b I=ijab



a $E[I(1) - 2I(2)]$ $E[I(1) - 2I(2)]$

b I=ijaa

a $E[I(1) - 2I(2)]$ $E[I(1) - 2I(2)]$

b I=ijbb

a

b --- $E[I(1) - 2I(2)]$

c I=ijac

WW Entries, ISEG= 4 & 5 6 , continued.

I=ijac E[I(1) - 2I(2)] E[I(3) - 2I(2)]

I=ijbc E[I(1) - 2I(2)] E[I(3)- 2I(2)]

I=ijbc --- E[I(1) - 2I(2)]

WW Entries, ISEG= 4 & 5 6, continued.

I=ijab

E[I(1) - 2I(2)] E[I(3) - 2I(2)]

I=ijab

--- E[I(1) - 2I(2)]

References

1. E. R. Davidson, D. Feller and P. Phillips, Chem. Phys. Lett. 76, 416 (1980).

2. G. Breit, Phys. Rev. 34, 553 (1929).

3a. J. P. Desclaux and P. Pyykko, Chem. Phys. Lett. 39, 300 (1976).

b. Y. S. Lee and A. D. McLean, to be published.

4a. H. Eyring, J. Walter and G. E. Kimball, Quantum Chemistry (John Wiley & Sons, Inc., New York, 1944).

b. M. Born and J. R. Oppenheimer, Ann. Physik 84, 457 (1927).

5a. R. L. Martin, Chem. Phys. Lett. 75, 290 (1980).

b. R. L. Martin and P. J. Hay, J. Chem. Phys. 75, 4539 (1981).

6a. D. R. Hartree, Proc. Cambridge Phil. Soc. 24, 89 (1928).

b V. Fock, Z. Physik 61, 126 (1930).

7. H. F. Schaefer, The Electronic Structure of Atoms and Molecules, (Addison-Wesley, Reading, 1972).

8a. L. B. Harding and W. A. Goddard, J. Chem. Phys. 67, 1777 (1977).

b. P. Saxe, H. F. Schaefer and N. C. Handy, J. Amer. Chem. Soc. 85, 745 (1981).

9. A. C. Wahl, J. Chem. Phys. 41, 2600 (1964).

10a. E. A. Hylleraas, Z. Physik 48, 469 (1928).

b. I. Shavitt, in Modern Theoretical Chemistry, edited by H. F. Schaefer (Plenum, New york, 1977), Vol. 3, pp. 189-275.

c. S. F. Boys, Proc. R. Soc. London, Ser. A 200, 542 (1950).

11a. B. Roos, Chem. Phys. Lett. 15, 153 (1972).

11b. B. Roos and P. E. M. Seigbahn, in Modern Theoretical Chemistry, edited by H. F. Schaefer (Plenum, New York, 1977), Vol. 3, pp. 277=318.

12a. A. Pipano and I. Shavitt, Intern. J. Quantum Chem. 2, 741 (1968).

b. J. Paldus, J. Cizek and I. Shavitt, Phys. Rev. A5, 50 (1972).

c. R. P. Hosteny, R. R. Gilman, T. H. Dunning, A. Pipano and I. Shavitt, Chem. Phys. Lett. 7, 325 (1970).

d. C. W. Baushlicher and I. Shavitt, J. Amer. Chem. Soc. 100, 739 (1978).

e. W. D. Laidig, P. Saxe and H. F. Schaefer, J. Chem. Phys. 73, 1765 (1980).

f. P. Saxe, H. F. Schaefer and N. C. Handy, Chem. Phys. Lett. 79, 202 (1981).

13. M. Moshinsky, Group Theory and the Many-Body Problem (Gordon & Breach, New York, 1968).

14a. J. Paldus, in The Unitary Group for the Evaluation of Electronic Energy Matrix Elements, "Lecture Notes in Chemistry", edited by J. Hinze (Springer, Berlin, 1981) Vol. 22, pp.1-50.

b. I. Shavitt, ibid, pp. 51-99.

c. G. W. F. Drake and M. Schlesinger, Phys. Rev. A 15, 1980 (1977).

d. J. Paldus and M. J. Boyle, Physica Scripta 21, 295 (1980).

e. I. Shavitt, "New Methods in Computational Chemistry and their Application on Modern Super-Computers", Report to NASA (Batelle, Columbus Labs., Columbus, Ohio, June 29 1979).

15a. A. Bunge, J. Chem. Phys. 53, 20 (1970).

b. A. D. McLean and B. Liu, J. Chem. Phys. 58, 1066 (1973).

c. B. R. Brooks and H. F. Schaefer, J. Chem. Phys. 70, 5092 (1979).

d. B. R. Brooks, W. D. Laidig, P. Saxe, N. C. Handy and H. F. Schaefer, Physica Scripta 21, 312 (1980).

16. E. R. Davidson, J. Comput. Phys. 17, 87 (1975).

17a. P. E. M. Seigbahn, J. Chem. Phys. 70, 5391 (1979).

b. P. E. M. Seigbahn, J. Chem. Phys. 72, 1647 (1980).

18. H. Lischka, R. Shepard, F. B. Brown and I. Shavitt, Int. J. Quantum Chem., Symp. 15, 91 (1981).

19. V. R. Saunders and J. H. van Lenthe, to be published.

20. P. Saxe, D. J. Fox, H. F. Schaefer and N. C. Handy, to be published.

21. T. H. Dunning, J. Chem. Phys. 53, 3823 (1970).

22. S. Huzinaga, J. Chem. Phys. 42, 1293 (1965).

23. P. O. Lowdin, Adv. Chem. Phys. 2, 207 (1959).

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720