Some Fortran–77 Programs for the Study of Large Amplitude Vibrations in Small Molecules

by

Kerry J. Davie

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Theoretical Chemistry

Winnipeg, Manitoba, 1988

ISBN 0-315-44161-5

SOME FORTRAN-77 PROGRAMS FOR THE STUDY OF LARGE AMPLITUDE VIBRATIONS IN SMALL MOLECULES

BY

KERRY J. DAVIE

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

This work presents three FORTRAN–77 programs which are applicable to the field of Large Amplitude Vibrations (LAV) of small molecules. Each program existed already in Hewlett Packard Basic and a translation was carried out with modifications in each case.

The first program produces a matrix which is used to generate Generalized Jacobi Vectors corresponding to a variety of orthonormalization procedures.

The second program generates a matrix of molecular potential energy values for various forms of the potential in various coordinate systems.

Finally, the third program solves the differential equation

$$[A(x)d^2/dx^2 + B(x)d/dx]\psi(x) = [V(x) - E]\psi(x) \ .$$

ACKNOWLEDGEMENTS

The author wishes to thank Dr. R. Wallace for his supervision and helpful discussions throughout the time of this work.

The author also wishes to express his appreciation to J. P. Leroy, a colleague in Dr. Wallace's research laboratory at the University of Manitoba, for the patient answering of many questions.

To Joanne

Table of Contents

List of Figures

Chapter 1

Introduction

The study of vibrations in N atom molecules has often been carried out in the context of normal [3] and local [1,2] mode models. It is useful to describe some of the features of the normal and local mode models. The normal mode model has three important characteristics [3]. These are:

a.) The metric tensor elements are assumed to be constants.

b.) Normal coordinates are orthogonal.

c.) The principal axes of the potential ellipsoid lie parallel to the normal–coordinate axes.

The normal mode model breaks down when large amplitude vibrations are considered, due to the violation of (a) and (b).

For both normal and local modes, the zero order Hamiltonian for nuclear motion, H_0, is a sum of independent one dimensional Hamiltonians [2]. The normal mode Hamiltonian is written

$$H_0^{nm}(\{\xi\}) = \sum_i \left(-(\hbar^2/2)G'_{ii}\frac{\partial^2}{\partial\xi_i^2} + k'_{ii}\xi_i^2\right) \qquad (1.1)$$

and the local mode zero order Hamiltonian can be written

$$H_o^{lm}(\{a_k\}) = \sum_i \left(-(\hbar^2/2)G_{ii}\frac{\partial^2}{\partial a_i^2}\, 2 + k_{ii}a_i^2 + k_{iii}a_i^3 + \ldots\right) . \qquad (1.2)$$

where ξ_i denotes the ith normal mode coordinate, and a_i denotes the ith local mode coordinate. The part of the Hamiltonian not included in (1.1) and (1.2) is considered as a perturbation, H^1, coupling the zero–order solutions, ψ_0, and preventing complete separability. The total Hamiltonian is $H = H^0 + H^1$. Local mode models, although being large amplitude models, suffer from the fact that they do not include, or systematically consider the bending motion of molecules, and they are not presented in a form suitable for including rovib interactions. Rovib interactions are important in relaxation mechanisms for highly excited vibrational states, a field of application for Large Amplitude Vibration (LAV) theory.

Wallace, in a series of papers [1,4], has treated the metric tensor elements as variables (Note the difference from the normal mode model.) and a variety of coordinate systems have been employed for the triatom. These coordinates are orthogonal for arbitrary amplitude. The form of the zero order Hamiltonian is analogous to that given in Eqns. 1.1 and 1.2, that is

$$H_0 = H(x) + H(y) + H(z) . \qquad (1.3)$$

The total Hamiltonian is

$$H = H_0 + V' \tag{1.4}$$

where V' is the non separable part of the potential and the total potential, V, is given by

$$V = V(x) + V(y) + V(z) + V' \tag{1.5}$$

V(x), V(y), and V(z) are the separable parts of V and are incorporated into H(x), H(y), and H(z), respectively. These papers present a method for determining that coordinate system in which the potential is most separable. Plots of V (with one of the coordinates held constant) are compared to plots of V(x) + V(y), V(y) + V(z), and V(x) + V(z). That coordinate system for which the plots agree in the largest energy range, is the system of choice.

In a more recent work by Wallace [5], a more general approach to nuclear motion was applied. A "family" of orthogonal internal cartesian coordinates was incorporated into the Hamiltonian framework prescribed by Hirschfelder et al [6]. In this framework, the Hamiltonian for relative motion is obtained by integrating over the Euler angles describing rotation.

In [5], orthogonal polar coordinates, related to the orthogonal cartesian coordinates, are used to derive a Hamiltonian, the zero order part of which has the form used in [1,4] for triatoms. The polar 'radial' coordinates are orthogonal to each other and all angular coordinates and the polar internal angular coordinates are orthogonal to each other but not to the Euler angles. The Hamiltonian in this case is

$$H = H_0 + T_1 + T_2 + V_c \qquad (1.6)$$

where V_c is the nonseparable part of the potential. T_1 represents the coupling of the rotational and internal motions, T_2 represents the mixing of the rotational states alone, and H_0 is the zero order Hamiltonian (3N–6 one dimensional operators). If the ground ('S') rotational state of the molecule is being considered, the Hamiltonian reduces to $H_0 + T_1 + V_c$.

Leroy and Wallace have now presented a form for the kinetic energy operator for relative motion of a group of particles in a general non–inertial reference frame [33]. The form of the Hamiltonians which result from specific frames employed, are analogous to those in Eqn. 1.6. The frames employed are defined in terms of Generalized Jacobi Vectors (GJVs) [4,23,33].

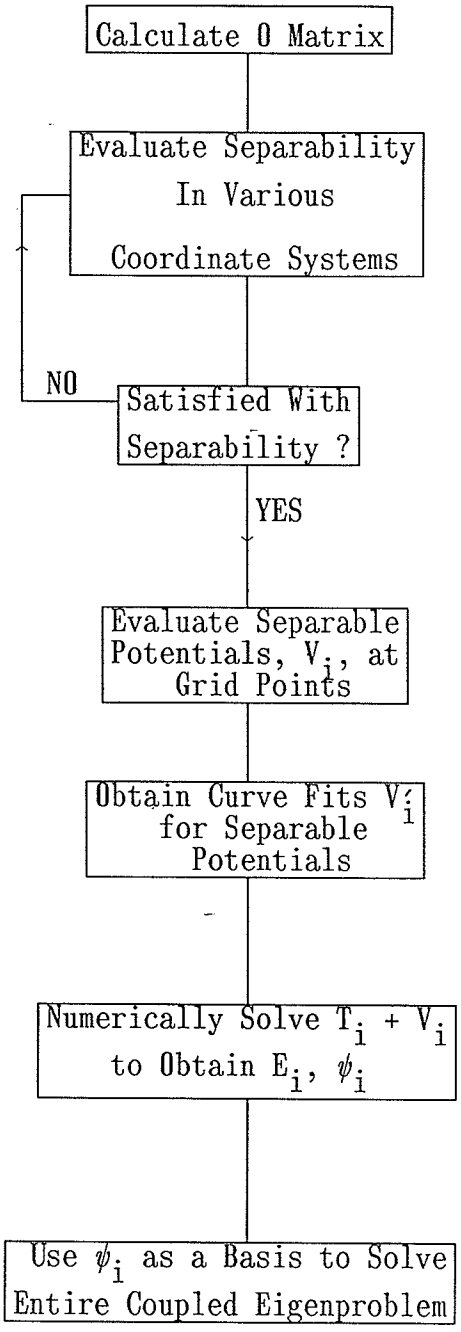
It is desirable to have available a variety of procedures which allow the determination of GJVs for N atom molecules. This has recently been accomplished [6]. The approach taken is to generate orthonormalization matrices, O, which convert from scalar bond distance–angle coordinates to scalar coordinates corresponding to a variety of Jacobi–type orthonormal coordinates. The generation of these matrices takes into account any symmetries inherent in the molecular hamiltonian.

The orthonormalization matrices, O, are used to transform the Gram matrix, G, of the scalar bond distance–angle coordinates to the Gram matrix, G′, of the scalar coordinates corresponding to a variety of GJVs as

$$OGO^t = G' \; . \qquad (1.7)$$

Once these coordinates have been obtained, further coordinate transformations to polar or hyperspherical coordinates can be performed.

As indicated earlier, the coordinate set of choice is that in which the potential is most separable. Once a particular coordinate system is decided upon, the separable potentials, V_i, are determined at a given number of grid points for a suitable range of each coordinate. Curve fits of this separable potential, V_i', are then calculated. Solution of the zero order Hamiltonian is carried out numerically to determine the E_{i0}'s and ψ_{i0}'s. The ψ_{i0}'s are used as a basis for a matrix representation of V_c and T_1 (see Eqn. 1.6). The entire procedure is given in the following flow chart.



This work is concerned with providing FORTRAN–77 code for three parts of the above procedure.

The first program, discussed in Chapter 1, allows the user to obtain O matrices by a variety of methods for molecules with two to

fourteen atoms. Parts of the molecule having symmetry can be treated separately from the rest of the molecule, with a global transformation produced by the end of the program. In Chapter 2, a program for obtaining contour plots of molecular potential energy functions for molecules is discussed. This program can handle three and four atom molecules, and allows the user to obtain plots of the total potential, or of the separable or non–separable parts. Chapter 3 is concerned with the numerical solution of one dimensional zero order Hamiltonians, which may or may not contain a first derivative term. More precisely, the solution of differential equations of the form

$$[A(x)d^2/dx^2 + B(x)d/dx]\psi(x) = [V(x)-E]\psi(x) \qquad (1.8)$$

by the re–normalized Numerov method with extensions [26] is considered. Finally, concluding comments and suggestions for further work are presented in Chapter 5. The source code for each program is presented in Appendices A to C. Some test programs are given in Appendix D.

Each of the afore mentioned programs was written in Micro–soft's FORTRAN–77 for an IBM AT personal computer. Batch files have been written to allow the user to start execution of each program with a minimum of effort.

Chapter 2

The O Matrix Generator Program

2.1. INTRODUCTION

This chapter presents the general background behind the program *GJVGEN* and gives some examples of results which can be obtained through use of the program.

In what follows, it is understood that the O matrix, or **O**, refers to transformation matrices which orthonormalize the relative basis of the contravariant metric tensor. These matrices are generated for the conversion of scalar bond distance–angle coordinates to scalar coordinates corresponding to a variety of Jacobi–type orthonormal coordinates [6]. Most of the theory behind the program *GJVGEN* is taken from reference 2.

2.2. THEORY OF THE ORTHONORMALIZATION (O) MATRIX

2.2.1. Center of Mass/Relative Basis For The Metric Tensor

The vector describing the configuration of an N atom molecule is written as

$$\mathbf{X} = \Sigma_i \, x^{i\alpha} \mathbf{e}_{i\alpha} \quad . \tag{2.1}$$

The orthogonal but not normal base vectors of the configuration space are

$$\mathbf{e}_{i\alpha} = \mathbf{e}_i \otimes \mathbf{d}_3 \, . \tag{2.2}$$

$\mathbf{e}_i$ are the orthogonal covariant base vectors, and $\mathbf{d}_3$ are the 'physical' three–dimensional–space base vectors. The $\mathbf{e}_i$ can be written in terms of orthonormal base vectors $\mathbf{e'}_i$:

$$\mathbf{e}_i = m_i^{1/2} \mathbf{e'}_i \quad . \tag{2.3}$$

This implies that

$$<\mathbf{e}_i,\mathbf{e}_i> = m_i \, . \tag{2.4}$$

The starting point in *GJVGEN* is the covariant metric tensor. This tensor has block diagonal form since its basis is $\mathbf{e}_i$ [6]. For a molecule

with N atoms, the covariant metric tensor, g, is

$$g = \begin{bmatrix} m_1 & 0 & 0 & 0 \ldots & & 0 \\ 0 & m_2 & 0 & 0 \ldots & & 0 \\ 0 & 0 & m_3 & 0 \ldots & & 0 \\ 0 & 0 & 0 & m_4 \; 0 \ldots & & 0 \\ \ldots & & & & & \\ & & & & & m_N \end{bmatrix} . \qquad (2.5)$$

The N masses of the molecule are read into the corresponding matrix **M** in *GJVGEN* along the main diagonal.

The bases of the contravariant metric tensor is denoted by $\tilde{e}$. The contravariant metric tensor, $\tilde{e} \cdot \tilde{e}^t$, is -

$$\tilde{g} = \begin{bmatrix} 1/m_1 & 0 & 0 & 0 \ldots & & 0 \\ 0 & 1/m_2 & 0 & 0 \ldots & & 0 \\ 0 & 0 & 1/m_3 & 0 \ldots & & 0 \\ 0 & 0 & 0 & 1/m_4 & & 0 \\ \ldots & & & & & \\ & & & & & 1/m_N \end{bmatrix} . \qquad (2.6)$$

The contravariant basis is orthogonal but not normalized.

The metric tensor, $\tilde{g}$, is now transformed into its center of mass/relative representation. The new basis, $\tilde{e}'$, is made up of vectors which are differences between the $\tilde{e}_i$, and one center of mass vector [6]. The difference vectors are defined as

$$\tilde{e}'_k = -\tilde{e}_i + \tilde{e}_j \quad . \tag{2.7}$$

The center of mass vector is constructed from the $\tilde{e}_i$ in the usual way [5]:

$$\tilde{e}_{CM} = 1/M\Sigma_i m_i \tilde{e}_i \tag{2.8}$$

where M is the total mass of the molecule. ($M = \Sigma_i m_i$) The matrix which transforms $\tilde{e}$ into $\tilde{e}'$ is Z. The form of Z is [6]

$$Z = \begin{matrix} \\ (k) \\ \\ CM \end{matrix} \begin{matrix} \begin{matrix} \quad (i) \qquad\qquad (j) \end{matrix} \\ \begin{bmatrix} \cdots & & & & & & \\ 0 & \cdots & -1 & \cdots & 1 & \cdots & 0 \\ \cdots & & & & & & \\ m_1/M & \cdots & m_i/M & \cdots & m_j/M & \cdots & m_N/M \end{bmatrix} \end{matrix} \quad . \tag{2.9}$$

Operating on the basis $\tilde{e}$ with Z gives the new basis,

$$Z\tilde{e} = \tilde{e}' \quad . \tag{2.10}$$

The new metric tensor, $\tilde{g}'$, is $\tilde{e}' \cdot \tilde{e}'^{t}$. From Eqn. 2.10, one obtains the following for $\tilde{g}'$:

$$\tilde{Z}\tilde{e} \cdot \tilde{e}^{t}\tilde{Z}^{t} = \tilde{g}' \quad , \tag{2.11}$$

$$\tilde{Z}\tilde{g}\tilde{Z}^{t} = \tilde{g}' \quad , \text{ and} \tag{2.12}$$

$$\tilde{g}' = \begin{bmatrix} \mu_1 & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1(N-1)} & 0 \\ \alpha_{21} & \mu_2 & \alpha_{23} & \cdots & \alpha_{2(N-1)} & 0 \\ \cdots & & & & & \\ \alpha_{(N-1)1} & \alpha_{(N-1)2} & \alpha_{(N-1)3} & \cdots & \mu_{N-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1/M \end{bmatrix}, \tag{2.13}$$

where $\mu_i = (m_a + m_b)/m_a m_b$, and m_a and m_b are the masses of the atoms connected by $\tilde{e}'_i$. $\alpha_{ij} = \pm 1/m_i$ if the base-vectors of $\tilde{g}'$, $\tilde{e}'_i$ and $\tilde{e}'_j$, 'share' a common atom or $\alpha_{ij} = 0$ if they do not. The center of mass base vector is orthogonal to the relative base vectors. The remainder of the program is concerned with the orthonormalization of the subspace describing relative motion.

The procedure outlined above is carried out for each molecule run on *GJVGEN*, with the user inputing the masses and topology of the molecule.

The base vectors ,$\tilde{e}'$, can be orthonormalized in a global sense, or they can be partitioned into groups in order that a 'pre–orthonormalization' can be performed on each group. As well, particular vectors of these groups can be orthonormalized. The methods

available for orthonormalization are i) Totally Symmetric, ii) Gram–Schmidt, and iii) Irreducible Symmetric.

2.2.2. Totally Symmetric Orthonormalization

2.2.2.1. Norm Dependent Symmetrization

$\tilde{\mathbf{g}}'$ is the contravariant metric tensor of the relative subspace with basis $\tilde{\mathbf{e}}'$. $\tilde{\mathbf{g}}'$ is written as $\tilde{\mathbf{e}}'\tilde{\mathbf{e}}'^{\mathrm{t}}$. Since any matrix that can be written in this way is positive definite [11], $\tilde{\mathbf{g}}'$ is positive definite.

The nonorthonormal basis $\tilde{\mathbf{e}}'$ is to be transformed into the orthonormal basis $\tilde{\mathbf{E}}$. Totally Symmetric orthonormalization is defined by

$$\mathbf{O} = \mathbf{O}^{\mathrm{t}} \,. \tag{2.14}$$

Then, since $\tilde{\mathbf{E}} = \mathbf{O}\tilde{\mathbf{e}}'$,

$$\tilde{\mathbf{E}}'\tilde{\mathbf{E}}'^{\mathrm{t}} = 1 = \mathbf{O}\tilde{\mathbf{e}}'\tilde{\mathbf{E}}'^{\mathrm{t}} \quad \text{and} \quad \tilde{\mathbf{E}}'^{\mathrm{t}} = \tilde{\mathbf{e}}'^{\mathrm{t}}\mathbf{O}^{\mathrm{t}} = \tilde{\mathbf{e}}'^{\mathrm{t}}\mathbf{O}, \tag{2.15}$$

which leads to

$$\mathbf{O}^{-1} = \tilde{\mathbf{e}}'\tilde{\mathbf{E}}'^{\mathrm{t}} \text{ and } \mathbf{O}^{-1} = \tilde{\mathbf{E}}'\tilde{\mathbf{e}}'^{\mathrm{t}} \,. \tag{2.16}$$

This means that $\tilde{\mathbf{E}}'\tilde{\mathbf{e}}'^{\mathrm{t}} = \tilde{\mathbf{e}}'\tilde{\mathbf{E}}'^{\mathrm{t}}$. $\mathbf{O}$ is determined from [5]

$$\mathbf{O} = \tilde{\mathbf{g}}'^{-1/2} \ . \tag{2.17}$$

Since $\tilde{\mathbf{g}}'$ is positive definite, $\tilde{\mathbf{g}}'^{-1}$ is also positive definite and its **unique positive** square root exists. It is given by [8]

$$\mathbf{O}_s = \lim_{n \to \infty} \mathbf{O}_n \ . \tag{2.18}$$

$$\mathbf{O}_0 = \mathbf{0}, \text{ and } \mathbf{O}_{n+1} = \mathbf{O}_n + (\tilde{\mathbf{g}}'^{-1} - \mathbf{O}_n^2)/2 \ . \tag{2.19}$$

$\mathbf{O}_s$ is a polynomial in the symmetric matrix $\tilde{\mathbf{g}}'^{-1}$, and therefore all $\mathbf{O}_n$, including $\mathbf{O}_s$, are symmetric. In practice, 470 iterations are performed, i.e. $n = 470$.

In certain pathological circumstances, Eqns. 2.18 and 2.19 fail to converge on the square root of $\tilde{\mathbf{g}}'^{-1}$. When this happens, $\tilde{\mathbf{g}}'^{1/2}$ is calculated, followed by the inversion.

2.2.2.2. Norm Independent Symmetrization

The nonorthonormal basis $\tilde{\mathbf{e}}'$ is to be transformed into the orthonormal basis $\tilde{\mathbf{E}}$. This is carried out by two consecutive transformations.

In the first, $\tilde{\mathbf{e}}'$ is transformed into the normalized basis $\tilde{\mathbf{p}}$ by the normalization matrix **N**:

$$\mathbf{N}\tilde{\mathbf{e}}' = \tilde{\mathbf{p}} \ . \tag{2.20}$$

In terms of the metric tensor, $\tilde{g}'$, the first transformation is

$$\tilde{N}\tilde{g}'\tilde{N}^t = \tilde{g}^* \quad . \tag{2.21}$$

$\tilde{g}^*$ has 1's on the main diagonal since the base vectors are normalized. Equivalent symmetric orthogonalization of $\tilde{p}$ is now performed as outlined in section 2.2.2.1. (i.e. $O_s = O_s{}^t$). This leads to

$$\tilde{E}\tilde{p}^t = \tilde{p}\tilde{E}^t. \tag{2.22}$$

This second transformation is identical with that in 2.2.2.1., except that $\tilde{g}^{*-1/2}$ is calculated, not $\tilde{g}'^{-1/2}$. The final O matrix, O_{final}, is

$$O_{final} = O_s N \; . \tag{2.23}$$

2.2.3. Gram–Schmidt Orthonormalization

2.2.3.1. General Gram–Schmidt Procedure

This procedure orthonormalizes the source base vectors ($\tilde{e}'_1$, $\tilde{e}'_2$, ...,$\tilde{e}'_k$) in stepwise manner [7].

The first vector is simply normalized.

$$\tilde{E}_1 = \tilde{e}'_1/|\tilde{e}'_1| \tag{2.24}$$

where $|\tilde{e'}_1|$ indicates the magnitude of $\tilde{e'}_1$.

$\tilde{E}_2$ is constructed from the component of $\tilde{e'}_2$ orthogonal to the space spanned by $\tilde{E}_1$:

$$\tilde{E}_2 = [\tilde{e'}_2 - < \tilde{e'}_2, \tilde{E}_1 > \tilde{E}_1]/|\tilde{e'}_2 - < \tilde{e'}_2, \tilde{E}_1 > \tilde{E}_1| \ . \qquad (2.25)$$

This procedure is continued until there are k orthonormal vectors. In each case, the vector $\tilde{E}_i$ is constructed by computing the component of $\tilde{e'}_i$, orthogonal to the space spanned by $\tilde{E}_{i-1}$, $\tilde{E}_{i-2}$, ...$\tilde{E}_1$.

2.2.3.2. Matrix Implementation

The orthonormal base vectors, $\tilde{E}_i$, may be written in terms of the non–orthonormal base vectors as [6]

$$\begin{aligned}
\tilde{E}_1 &= D_1^{-1/2}\tilde{e'}_1 \\
\tilde{E}_2 &= \Delta_2(D_1D_2)^{-1/2} \\
&= \begin{vmatrix} \tilde{e'}_1\cdot\tilde{e'}_1 & \tilde{e'}_1\cdot\tilde{e'}_2 \\ \tilde{e'}_1 & \tilde{e'}_2 \end{vmatrix} (D_1D_2)^{-1/2} , \\
\tilde{E}_k &= \Delta_k(D_{k-1}D_k)^{-1/2} \\
&= (D_{k-1}D_k)^{-1/2} \begin{vmatrix} \tilde{e'}_1\cdot\tilde{e'}_1 & \tilde{e'}_2\cdot\tilde{e'}_1 & \tilde{e'}_k\cdot\tilde{e'}_1 \\ \cdots & & \\ \tilde{e'}_1\cdot\tilde{e'}_{k-1} & \tilde{e'}_2\cdot\tilde{e'}_{k-1} & \tilde{e'}_k\cdot\tilde{e'}_{k-1} \\ \tilde{e'}_1 & \tilde{e'}_2 & \tilde{e'}_k \end{vmatrix}
\end{aligned} \qquad (2.26)$$

D_k is the Gram determinant formed from $\tilde{e}'_1$, ...,$\tilde{e}'_k$. The determinant Δ_k is expanded in terms of the co–factors, c_{ki}, of the last row. In this way, for each $\tilde{E}_i$, the orthonormalization constants for the $\tilde{e}'_i$ are obtained from $c_{ki}(D_{k-1}D_k)^{-1/2}$.

The above procedure is contained in the two subroutines *GRAMSC* and *MINORS* in the program *GJVGEN.*

2.2.3.2.1. *S*ubroutine *Gramsc*

The matrix SEQNCE is assigned values according to the order of orthonormalization. For example, if the molecule has 4 atoms and the order of global orthonormalization of the three nonorthogonal base vectors is: $\tilde{e}'_3$, $\tilde{e}'_2$, $\tilde{e}'_1$, then

$$\mathbf{SEQNCE} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} . \qquad (2.27)$$

The metric tensor of the contravariant basis, *GRAM*, is transformed as

$$[\mathbf{SEQNCE}][\mathbf{GRAM}][\mathbf{SEQNCE}]^{-1} = \mathbf{GPERM} . \qquad (2.28)$$

The inverse of **SEQNCE** is used here. **SEQNCE** is an orthogonal matrix,

that is, $[\mathbf{SEQNCE}]^{-1} = [\mathbf{SEQNCE}]^{t}$. The original metric tensor has the following form:

$$\mathbf{GRAM} = \begin{bmatrix} \tilde{g}'_{11} & \tilde{g}'_{12} & \tilde{g}'_{13} \\ \tilde{g}'_{21} & \tilde{g}'_{22} & \tilde{g}'_{23} \\ \tilde{g}'_{31} & \tilde{g}'_{32} & \tilde{g}'_{33} \end{bmatrix} , \qquad (2.29)$$

where $\tilde{e}'_i$ are the nonorthogonal source vectors. The form of **GPERM** is (where Eqn. 2.27 has been used)

$$\mathbf{GPERM} = \begin{bmatrix} \tilde{g}'_{33} & \tilde{g}'_{32} & \tilde{g}'_{31} \\ \tilde{g}'_{23} & \tilde{g}'_{22} & \tilde{g}'_{21} \\ \tilde{g}'_{13} & \tilde{g}'_{12} & \tilde{g}'_{11} \end{bmatrix} . \qquad (2.30)$$

In line 41 of *GRAMSC*, D_1 is assigned to **DETERM**(2), the vector containing the D_k's. Note that due to the transformation of **GRAM** by **SEQNCE**, **GPERM**(1,1) is **always** the square of the magnitude of the first base vector to be normalized.

In the DO loop starting at line 42 of *GRAMSC*, the values of the D_k's are assigned to the vector **DETERM**. The D_k's are found by calculating the determinant of the K X K sub–matrix of **GPERM**. K is the index of the DO loop. Each time through the loop, the subroutine

MINORS is called, and the co–factors of the last row of Δ_k are calculated. These values are assigned to the first K columns of the Kth row of O.

In line 62, the normalization constant for the highest priority source vector is calculated. In the DO loop starting on line 63, the final orthonormalization constants of the lower priority vectors are calculated and assigned to rows 2 to N–1 (there are N atoms in the molecule.) of the O matrix. In the DO loop starting on line 64, the K orthonormalization constants for $\tilde{E}_k$ are calculated by multiplying the cofactors of Δ_K by $(D_{k-1}D_k)^{-1/2}$.

The last three lines of *GRAMSC* rearrange the rows of O so that row one corresponds to $\tilde{E}_1$, row two corresponds to $\tilde{E}_2$, etc. This is done by carrying out the inverse of the transformation that was done on the metric tensor, **GRAM**. This transformation is $[\mathrm{SEQNCE}]^{-1}\mathrm{O}[\mathrm{SEQNCE}]$. For the example of a four atom molecule discussed above, O would have the form

$$\mathbf{O} = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix} . \qquad (2.31)$$

The final form of **O** after the transformation would be,

$$\mathbf{0} = \begin{bmatrix} f & e & d \\ 0 & c & b \\ 0 & 0 & a \end{bmatrix} . \qquad (2.32)$$

2.2.3.2.2. Subroutine *Minors*

This subroutine calculates the co–factors of the Kth (last) row of Δ_K. In the two DO loops starting on lines 4 and 6, DETMAT is assigned the values of GPERM (the metric tensor). The DO loop starting on line 22 deletes (assigns zero to) the Kth row of DETMAT. The DO loop starting on line 32 deletes column A (the column containing $\tilde{e}'_A$). A is the index of the outer DO loop and is incremented from 1 to K. At this point, the first K–1 rows of DETMAT are identical to Δ_K (see Eqn. 2.26). Finally, the position of $\tilde{e}'_A$ is given the value 1.0, since the co–factor of $\tilde{e}'_A$ is to be calculated. The determinant of DETMAT with the appropriate columns and rows deleted is then calculated in line 44. This is the co–factor of $\tilde{e}'_A$. The value of the co–factor is assigned to the Kth row and Ath column of O in line 46. The above algorithm is then performed k–1 more times to calculate the co–factors of the last row of Δ_k.

2.2.4. Irreducible Symmetric Orthonormalization

This transformation is derived from the eigenvalue equation [12]

$$\tilde{g'}T = T\Lambda \tag{2.33}$$

where T is the matrix whose columns are the eigenvectors, $\tilde{v}_i$, of $\tilde{g'}$ ($T = [\tilde{v}_1 \tilde{v}_2 ... \tilde{v}_N]$), and Λ is the diagonal matrix with the eigenvalues of $\tilde{g'}$ on the main diagonal. T is an orthogonal matrix [7]. Multiplication by T^{-1} on the left in Eqn. 2.33 gives

$$T^{-1}\tilde{g'}T = \Lambda = T^t\tilde{g'}T \ . \tag{2.34}$$

Clearly T is an orthogonalizing transformation matrix. To normalize Λ, pre and post multiplication by $\Lambda^{-1/2}$ is carried out.
This gives

$$\Lambda^{-1/2}T^t\tilde{g'}T\Lambda^{-1/2} = 1 \ . \tag{2.35}$$

The final O matrix is therefore $\Lambda^{-1/2}T^t$.

Implementation of the above in *GJVGEN* merely requires the calculation of the eigenvalues and eigenvectors of $\tilde{g'}$, a symmetric N x N matrix. Any subroutine which does this is called by the user from the subroutine *IRRSYM* (see Appendix A).

2.2.5. Global Orthonormalization

To describe this procedure, it is instructive to work through an example. The molecule dichloromethane (CH_2Cl_2) is chosen, and the atoms are numbered as follows: $C_1H_2H_3Cl_4Cl_5$.
The contravariant metric tensor is then

$$\tilde{g} = \begin{bmatrix} 1/12 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/36.0 & 0 \\ 0 & 0 & 0 & 0 & 1/36.0 \end{bmatrix} . \qquad (2.36)$$

The metric tensor is now transformed into its center of mass/relative representation. For dichloromethane, the new basis is $\tilde{e}' = (\tilde{e}'_1, \tilde{e}'_2, \tilde{e}'_3, \tilde{e}'_4, \tilde{e}'_{CM})^t$ where $\tilde{e}'_1 = \tilde{e}_2 - \tilde{e}_1$, $\tilde{e}'_2 = \tilde{e}_3 - \tilde{e}_1$, ... $\tilde{e}'_{CM} = m_1/M\tilde{e}_1 + m_2/M\tilde{e}_2 + m_3/M\tilde{e}_3 + m_4/M\tilde{e}_4 + m_5/M\tilde{e}_5$. M is the total mass of the molecule ($M = m_1 + m_2 + \ldots m_5$). The matrix which transforms $(\tilde{e}_1, \tilde{e}_2, \tilde{e}_3, \tilde{e}_4, \tilde{e}_5)$ into $\tilde{e}'$ is

$$Z = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ m_1/M & m_2/M & m_3/M & m_4/M & m_5/M \end{bmatrix} \qquad (2.37)$$

Operating on the basis $(\tilde{e}_1, \tilde{e}_2, ...,\tilde{e}_5)$ with Z gives the new basis,

$$\tilde{Ze} = \tilde{e}' \quad . \qquad (2.38)$$

The new metric tensor, $\tilde{g}'$, is $\tilde{e}' \cdot \tilde{e}'^t$. From Eqn. 2.11, one obtains the following for $\tilde{g}'$

$$\tilde{Z}\tilde{g}Z^t = \tilde{g}' \; . \qquad (2.39)$$

$$\tilde{g}' = \begin{bmatrix} \mu & 1/m_1 & 1/m_1 & 1/m_1 & 0 \\ 1/m_1 & \mu & 1/m_1 & 1/m_1 & 0 \\ 1/m_1 & 1/m_1 & \mu' & 1/m_1 & 0 \\ 1/m_1 & 1/m_1 & 1/m_1 & \mu' & 0 \\ 0 & 0 & 0 & 0 & 1/M \end{bmatrix} \qquad (2.40)$$

where $\mu = (m_1 + m_2)/m_2m_1$ and $\mu' = (m_1 + m_4)/m_4m_1$. The center of mass base vector is orthogonal to the relative base vectors. Attention is

now restricted to the non–orthonormal relative base vectors $\tilde{e}'_1, \ldots, \tilde{e}'_4$.

GJVGEN allows the user to choose either the totally symmetric, Gram Schmidt, or Irreducible Symmetric orthonormalization procedures (see sections **2.2.2.**, **2.2.3.**, **2.2.4.**, respectively).

By the nature of the Gram Schmidt procedure, any symmetry in $\tilde{e}'$ will be lost. The totally symmetric or irreducible symmetric procedures might be chosen.

For **norm dependent symmetrization**, the final O matrix is $\tilde{g}'^{-1/2}$.

$$\mathbf{O} = \tilde{g}'^{-1/2} = \begin{bmatrix} .97823 & -.02177 & -.12441 & -.12441 \\ -.02177 & .97823 & -.12441 & -.12441 \\ -.12441 & -.12441 & 4.2040 & -1.7959 \\ -.12441 & -.12441 & -1.79593 & 4.2040 \end{bmatrix} \tag{2.41}$$

The transformation of $\tilde{g}'$ $(\mathbf{O}\tilde{g}'\mathbf{O}^t)$ gives the matrix **1**.

The final orthonormal base vectors are

$$\begin{aligned} \tilde{E}_1 &= .97823\tilde{e}'_1 - .02177\tilde{e}'_2 - .12441\tilde{e}'_3 - .12441\tilde{e}'_4 \\ &\ldots \\ \tilde{E}_4 &= -.12441\tilde{e}'_1 - .12441\tilde{e}'_2 - 1.79593\tilde{e}'_3 + 4.20395\tilde{e}'_4 . \end{aligned} \tag{2.42}$$

In this case, all four orthonormal base vectors are symmetric upon interchange of the two hydrogens, and/or the two chlorines.

2.2.6. Orthonormalization by Partitioning Vectors Into Groups

2.2.6.1. Equivalent Symmetric

The base vectors, $\tilde{e}'$, of $\tilde{g}'$ fall clearly into two groups; A = ($\tilde{e}'_1$, $\tilde{e}'_2$), and B = ($\tilde{e}'_3$, $\tilde{e}'_4$). Each of these groups can be orthonormalized separately. If A and B are both orthonormalized by the symmetric procedure (see section 2.2.2.), the result is

$$\mathbf{O}_A = \begin{bmatrix} .963 & -.037 \\ -.037 & .963 \end{bmatrix}$$
$$\mathbf{O}_B = \begin{bmatrix} 4.112 & -1.846 \\ -1.846 & 4.112 \end{bmatrix} . \qquad (2.43)$$

O_A and O_B are combined into another O matrix (4 x 4) as follows.

$$\mathbf{O}_{Total} = \begin{bmatrix} \mathbf{O}_A & 0_2 \\ 0_2 & \mathbf{O}_B \end{bmatrix} \qquad (2.44)$$

0_2 is the 2 x 2 zero matrix. $[O_{Total}\tilde{e}']$ will give the new basis $\tilde{e}''$. The new metric tensor is given by

$$\tilde{g}'' = \tilde{e}''\tilde{e}''^{t} . \tag{2.45}$$

The transformation is written as

$$[O_{Total}\tilde{e}'][O_{Total}\tilde{e}']^{t} = O_{Total}\tilde{e}'\tilde{e}'^{t}O_{Total}^{t} \tag{2.46}$$

$$O_{Total}\tilde{g}'O_{Total}^{t} = \tilde{g}'', \tag{2.47}$$

where

$$\tilde{g}'' = \begin{bmatrix} 1 & 0 & A & A \\ 0 & 1 & A & A \\ A & A & 1 & 0 \\ A & A & 0 & 1 \end{bmatrix} \tag{2.48}$$

and A = 0.175 . Clearly, $\tilde{e}''_1$ and $\tilde{e}''_2$ are mutually orthogonal and normalized as are $\tilde{e}''_3$ and $\tilde{e}''_4$. However, $\tilde{e}''_1$ and $\tilde{e}''_2$ are not orthogonal to $\tilde{e}''_3$ and $\tilde{e}''_4$. The symmetric procedure is chosen to generate four orthonormal vectors, $\tilde{E}_1$, $\tilde{E}_2$, $\tilde{E}_3$, and $\tilde{E}_4$. The result is

$$\mathbf{O}' = \begin{bmatrix} 1.02525 & .02525 & -.09492 & -.09492 \\ .02525 & 1.02525 & -.09492 & -.09492 \\ -.09492 & -.09492 & 1.02525 & .02525 \\ -.09492 & -.09492 & .02525 & 1.02525 \end{bmatrix} . \qquad (2.49)$$

Application of $\mathbf{O}'$ to $\tilde{\mathbf{g}}''$ as $\mathbf{O}'\tilde{\mathbf{g}}''\mathbf{O}'^{t}$ yields $1 = \tilde{\mathbf{E}}\tilde{\mathbf{E}}^{t}$. The total transformation of the basis $\tilde{\mathbf{e}}'$ is $\mathbf{O}'\mathbf{O}_{\text{Total}} = \mathbf{O}$, i.e.

$$\mathbf{O}'\mathbf{O}_{\text{Total}} = \mathbf{O} = \begin{bmatrix} 0.986 & -.014 & -.215 & -.215 \\ -.014 & 0.986 & -.215 & -.215 \\ -.0879 & -.0879 & 4.16922 & -1.78878 \\ -.0879 & -.0879 & -1.78878 & 4.16922 \end{bmatrix} . \qquad (2.50)$$

The final orthonormal base vectors are

$$\begin{aligned} \tilde{\mathbf{E}}_1 &= 0.986\tilde{\mathbf{e}}'_1 - 0.014\tilde{\mathbf{e}}'_2 - .215\tilde{\mathbf{e}}'_3 - .215\tilde{\mathbf{e}}'_4 . \\ &\cdots \\ \tilde{\mathbf{E}}_4 &= -.0879\tilde{\mathbf{e}}'_1 - .0879\tilde{\mathbf{e}}'_2 - 1.78878\tilde{\mathbf{e}}'_3 + 4.16922\tilde{\mathbf{e}}'_4 . \end{aligned} \qquad (2.51)$$

Again, note that the final base vectors are symmetric upon interchange of the two hydrogen atoms and/or the two chlorine atoms.

2.2.6.2. Irreducible Symmetric

In this case, the group A and B vectors are separately orthonormalized by the irreducible symmetric procedure.

$$\mathbf{O}_A = \begin{bmatrix} -.707 & .707 \\ -.655 & -.655 \end{bmatrix}$$
$$\mathbf{O}_B = \begin{bmatrix} -4.213 & 4.213 \\ -1.604 & -1.604 \end{bmatrix} . \qquad (2.52)$$

$\mathbf{O}_A$ and $\mathbf{O}_B$ are again combined to give

$$\mathbf{O}_{Total} = \begin{bmatrix} \mathbf{O}_A & \mathbf{O}_2 \\ \mathbf{O}_2 & \mathbf{O}_B \end{bmatrix} . \qquad (2.53)$$

The transformation of $\mathbf{g}'$ by $\mathbf{O}_{Total}$,

$$\mathbf{O}_{Total}\tilde{\mathbf{g}}'\mathbf{O}_{Total}^{t} = \tilde{\mathbf{g}}'', \qquad (2.54)$$

gives $\tilde{\mathbf{g}}''$, where

$$\tilde{g}'' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \gamma \\ 0 & 0 & 1 & 0 \\ 0 & \gamma & 0 & 1 \end{bmatrix} \qquad (2.55)$$

and $\gamma = .350$. The new basis is $\tilde{e}''$. Clearly, $\tilde{e}''_1$ and $\tilde{e}''_2$ are mutually orthogonal and normalized as are $\tilde{e}''_3$ and $\tilde{e}''_4$. In fact, the only non–orthogonal pair are $\tilde{e}''_2$ and $\tilde{e}''_4$. If the Gram–Schmidt procedure is chosen to generate four orthonormal vectors, $\tilde{E}_1$, $\tilde{E}_2$, $\tilde{E}_3$, and $\tilde{E}_4$, and $\tilde{e}''_4$ is the last vector orthonormalized, the new transformation matrix is:

$$\mathbf{O}' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -.3731 & 0 & 1.0673 \end{bmatrix} . \qquad (2.56)$$

Application of $\mathbf{O}'$ to $\tilde{g}''$ as $\mathbf{O}'\tilde{g}''\mathbf{O}'^{t}$ yields $1 = \tilde{E}\tilde{E}^{t}$. The total transformation of the basis $\tilde{e}'$ is $\mathbf{O}'\mathbf{O}_{\text{Total}} = \mathbf{O}$, i.e.

$$\mathbf{0} = \mathbf{0}'\mathbf{0}_{\text{Total}} = \begin{bmatrix} -.707 & .707 & 0 & 0 \\ -.655 & -.655 & 0 & 0 \\ 0 & 0 & -4.213 & 4.213 \\ .245 & .245 & -1.712 & -1.712 \end{bmatrix} . \qquad (2.57)$$

The final orthonormal base vectors are

$$\begin{aligned} \tilde{E}_1 &= -.707\tilde{e}'_1 + .707\tilde{e}'_2 \\ &\cdots \\ \tilde{E}_4 &= .245\tilde{e}'_1 + .245\tilde{e}'_2 - 1.712\tilde{e}'_3 - 1.712\tilde{e}'_4. \end{aligned} \qquad (2.58)$$

In this case $\tilde{E}_1$ and $\tilde{E}_2$ are linear combinations of only $\tilde{e}'_1$ and $\tilde{e}'_2$, and $\tilde{E}_3$ is a linear combination of only $\tilde{e}'_3$ and $\tilde{e}'_4$. This is a desirable result, in that $\tilde{E}_1$ and $\tilde{E}_2$ represent orthonormalized carbon–hydrogen vectors with no mixing in of the carbon–chlorine vectors, and $\tilde{E}_3$ represents an orthonormalized carbon–chlorine vector with no mixing in of the carbon–hydrogen vectors. Unfortunately, the same obviously can not be said about $\tilde{E}_4$. As well, it can be seen that $\tilde{E}_1$ is anti–symmetric with respect to inter–change of the two hydrogens, and $\tilde{E}_3$ is anti–symmetric with respect to inter–change of the two chlorines. $\tilde{E}_2$ and $\tilde{E}_4$ are symmetric with respect to inter–change of the hydrogens and/or the chlorines. In order to reduce the mixing of the hydrogen and chlorine vectors, four new orthonormal vectors have been derived which do not possess the symmetry of the molecule.

2.2.7. Orthonormalization of Particular Vectors

After the program is finished with each group of vectors, the user is presented with the option of orthonormalizing any particular vectors. These vectors may be from any group of vectors. For example, for CH_2Cl_2, the user may be presented with the metric tensor given by Eqn. 2.55.

It can be seen that vectors two and four are not mutually orthogonal. The user inputs $\begin{matrix}4\\2\end{matrix}$ or $\begin{matrix}2\\4\end{matrix}$ to the prompts for the vectors to be orthonormalized. The program then performs a permutation of the vectors. This is done to ensure that the vectors to be orthonormalized have the highest priority in the metric tensor. Orthonormalization of the metric tensor is restricted to the upper left Q dimensional subspace, where Q is the number of particular vectors to be orthonormalized.

For the above case, after permutation, the metric tensor becomes

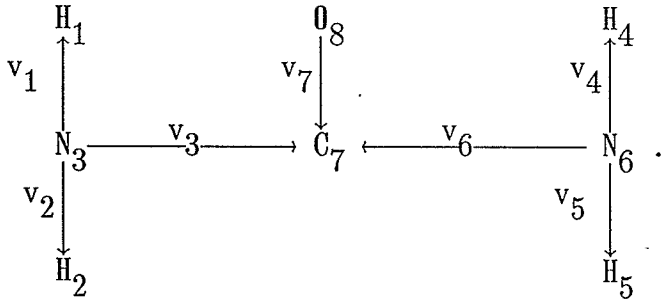
$$\tilde{g}''^{*} = \begin{bmatrix} 1.0 & \gamma & 0.0 & 0.0 \\ \gamma & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} . \qquad (2.59)$$

The program prompts the user for the choice of orthonormalization procedure for the Q vectors. For any choice of orthonormalization procedure, the above matrix would become I_4.

Use of the above part of the program that allows the user to orthonormalize any set of vectors can lead to interesting results.

To illustrate this, two very similar **O** matrices for urea are determined, the second one using the orthonormalization of particular vectors.

The molecule is numbered as shown (inter atom vectors v_i are indicated.):



For the first case, the vectors are partitioned into three groups. In terms of the contravariant bases, $\tilde{e}'$, group 1 contains $\tilde{e}'_1$, $\tilde{e}'_2$ and $\tilde{e}'_3$, group 2 contains $\tilde{e}'_4$, $\tilde{e}'_5$, and $\tilde{e}'_6$, and group 3 consists only of $\tilde{e}'_7$. Groups 1 and 2 are orthonormalized separately by the irreducible symmetric procedure, and group 3 is not normalized.

The six new normalized vectors and $\tilde{e}'_7$ are now orthonormalized by the Gram–Schmidt procedure. The order of orthonormalization of the seven vectors is 2, 5, 1, 4, 3, 6, and 7. The final O matrix is given in Fig. 2.1.

For the second case, the vectors are partitioned into five groups.

FIGURE 2.1: O MATRIX FOR UREA (CASE 1)

FIGURE 2.2: O MATRIX FOR UREA (CASE 2)

$$\begin{bmatrix} -.187 & -.187 & 2.617 & .000 & .000 & .000 & .000 \\ -.707 & .707 & .000 & .000 & .000 & .000 & .000 \\ -.652 & -.652 & -.139 & -.006 & -.006 & .079 & .000 \\ .130 & .130 & -1.819 & -.228 & -.228 & 3.187 & .000 \\ .000 & .000 & .000 & -.707 & .707 & .000 & .000 \\ -.005 & -.005 & .080 & -.652 & -.652 & -.139 & .000 \\ .078 & .078 & -1.246 & .078 & .078 & -1.246 & 3.425 \end{bmatrix}$$

FIG. 2.1

$$\begin{bmatrix} -.191 & -.191 & 2.616 & .000 & .000 & .000 & .000 \\ -.707 & .707 & .000 & .000 & .000 & .000 & .000 \\ -.650 & -.650 & -.162 & -.007 & -.007 & .092 & .000 \\ .133 & .133 & -1.817 & -.233 & -.233 & 3.185 & .000 \\ .000 & .000 & .000 & -.707 & .707 & .000 & .000 \\ -.006 & -.006 & .093 & -.650 & -.650 & -.162 & .000 \\ .078 & .078 & -1.246 & .078 & .078 & -1.246 & 3.425 \end{bmatrix}$$

FIG. 2.2

Group 1 contains $\tilde{e}'_1$, and $\tilde{e}'_2$, group 2 contains $\tilde{e}'_3$, group 3 contains $\tilde{e}'_4$ and $\tilde{e}'_5$, group 4 contains $\tilde{e}'_6$, and group 5 contains $\tilde{e}'_7$.

Groups 1 and 3 are orthonormalized separately by the totally symmetric procedure, and the other groups are not orthonormalized. Then, using the orthonormalization of particular vectors, vectors one, two, and three are orthonormalized by the irreducible symmetric procedure. Vectors four, five, and six are also orthonormalized in the same way. Finally, the six new vectors and $\tilde{e}'_7$ are orthonormalized by the Gram–Schmidt procedure in the same order as was used in case one. The final O matrix (Fig. 2.2) is very similar, but not identical to the one calculated in case one.

2.3. O MATRIX PROGRAM

The preceeding sections have made reference to the program *GJVGEN*. This program was originally written in HP Basic by J. P. Leroy and R. Wallace. A translation was made of the original program into Microsoft Fortran 77 . The program was then expanded to give it the capability to handle irreducible symmetric orthonormalization, and partitioning of a molecule's vectors into groups. As well, the 'particular vectors' feature was added. For more information on how to use *GJVGEN*, the reader is referred to the program's manual [9].

Chapter 3

The Plotting Program

3.1. INTRODUCTION

The program *NPLOT* is designed to allow the user to obtain contour plots of potential energy functions for three and four atom molecules in various coordinate systems. All coordinates of the particular system are fixed, except for two chosen by the user. The user has the option of plotting the total, separable or non–separable parts of the potential. This allows the user to determine that coordinate system in which the molecule's total potential is most separable. Complete separability is, of course, an unattainable ideal.

NPLOT was adapted from a program originally written in H. P. Basic by R. Wallace. The program was initially translated into Microsoft Fortran with a few changes. Later, more changes were made to make the program easier to use.

3.2. TRANSFORMATION OF BOND DISTANCES <—> BOND DISTANCE–BOND ANGLES

The analytical potential functions used (section 6.) are given as functions of the 3N–6 bond distances, q_i, of the N atom molecule. Bond vectors are written as $\vec{q}_i$. The first step in the transformation to generalized Jacobi coordinates is the conversion to bond distance–bond angle coordinates.

3.2.1. Three Atom Molecules

The transformation has the form [15]

$$\Phi_{12} = \arccos\left[\frac{(q_1)^2 + (q_2)^2 - (q_3)^2}{2q_1q_2}\right] . \qquad (3.1)$$

$\vec{q}_1$ and $\vec{q}_2$ are shown in Fig. 3.1 . A simple manipulation of 3.1 gives

$$q_3 = [(q_1)^2 + (q_2)^2 - 2q_1q_2\cos\Phi_{12}]^{1/2} \qquad (3.2)$$

as the inverse transformation.

FIGURES 3.1 AND 3.2: BOND DISTANCE BOND ANGLE COORDINATES FOR THE THREE BODY AND BRANCHED FOUR BODY

$\vec{q}_i$ are the bond vectors.

Φ_{ij} are the bond angles.

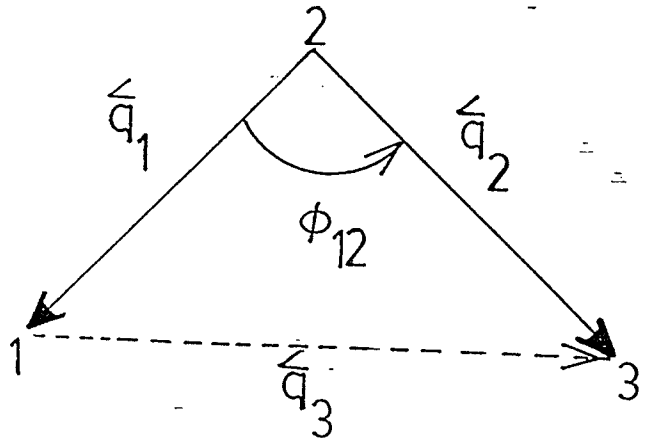

2
$\vec{q}_1$
$\vec{q}_2$
Φ_{12}
1
$\vec{q}_3$
3

FIG. 3.1

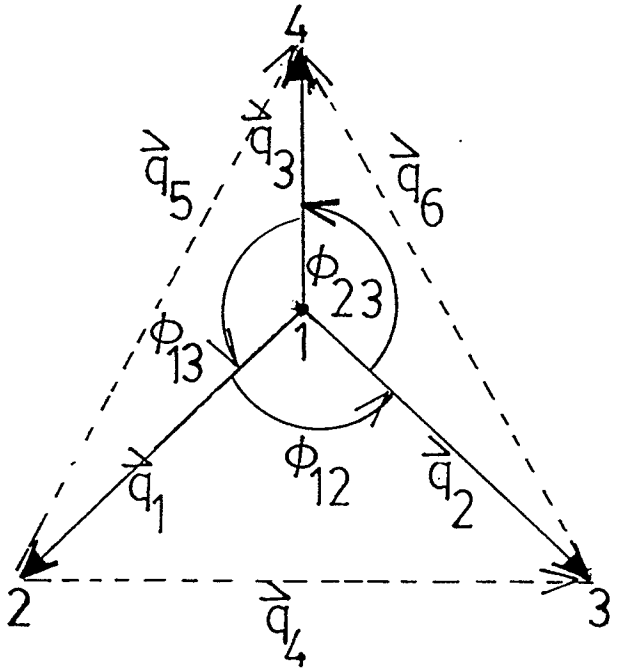

4
$\vec{q}_3$
$\vec{q}_5$
$\vec{q}_6$
Φ_{23}
Φ_{13}
1
$\vec{q}_1$
Φ_{12}
$\vec{q}_2$
2
$\vec{q}_4$
3

FIG. 3.2

3.2.2. Four Atom Molecules

There are two distinct types of four atom molecules, branched and non–branched. Both types can be considered as three different three atom problems.

3.2.2.1. Branched

The molecule is shown in Fig. 3.2. The three scalar bond distances and three bond angles needed for the transformation to Jacobi coordinates are q_1, q_2, and q_3 and Φ_{12}, Φ_{13}, and Φ_{23}, respectively. The angles are calculated analogously to section 3.2.1. by Eqns. 3.3 to 3.5 [7].

$$\Phi_{12} = \arccos\left[\frac{(q_1)^2 + (q_2)^2 - (q_4)^2}{2q_1q_2}\right] \tag{3.3}$$

$$\Phi_{13} = \arccos\left[\frac{(q_1)^2 + (q_3)^2 - (q_5)^2}{2q_1q_3}\right] \tag{3.4}$$

$$\Phi_{23} = \arccos\left[\frac{(q_2)^2 + (q_3)^2 - (q_6)^2}{2q_2q_3}\right] \tag{3.5}$$

The other three bond distances can be recovered by using Eqns. 3.6 to 3.8.

$$q_4 = [(q_1)^2 + (q_2)^2 - 2q_1q_2\cos\Phi_{12}]^{1/2} \quad (3.6)$$

$$q_5 = [(q_1)^2 + (q_3)^2 - 2q_1q_3\cos\Phi_{13}]^{1/2} \quad (3.7)$$

$$q_6 = [(q_2)^2 + (q_3)^2 - 2q_2q_3\cos\Phi_{23}]^{1/2} \quad (3.8)$$

3.2.2.2. Non–Branched

The molecule is shown in Fig. 3.3. The three bond distances needed are again q_1, q_2, and q_3. The angles between these bonds are calculated in Eqns. 3.9 to 3.11.

$$\Phi_{12} = \arccos\left[\frac{(q_1)^2 + (q_2)^2 - (q_4)^2}{2q_1q_2}\right] \quad (3.9)$$

$$\Phi_{13} = \pi - \arccos\left[\frac{(q_1)^2 + (q_3)^2 - (q_5)^2}{2q_1q_3}\right] \quad (3.10)$$

$$\Phi_{23} = \arccos\left[\frac{(q_4)^2 + (q_5)^2 - (q_1)^2 - (q_6)^2}{2q_2q_3}\right] \quad (3.11)$$

The expression for Φ_{23}, the angle between $\vec{q}_2$ and $\vec{q}_3$, is arrived at in the following way (see Figs. 3.3 and 3.4). Note that in Fig. 3.3 $\Phi'_{13} = \pi - \Phi_{13}$ is denoted.

$\vec{q}_3$ can be written as $\vec{q}_5 - \vec{q}_1$, and, from the dot product of $\vec{q}_3$ and $\vec{q}_2$, the following expression for $\cos\Phi_{23}$ is determined:

FIGURE 3.3: BOND DISTANCE BOND ANGLE COORDINATES FOR THE LINEAR FOUR BODY

$\vec{q}_i$ are the bond vectors.
Φ_{ij} are the bond angles.

FIGURE 3.4: LINEAR FOUR BODY SHOWING Φ_{25} USED IN THE DERIVATION OF Φ_{23}

$\vec{q}_i$ are the bond vectors.
Φ_{ij} are the bond angles.

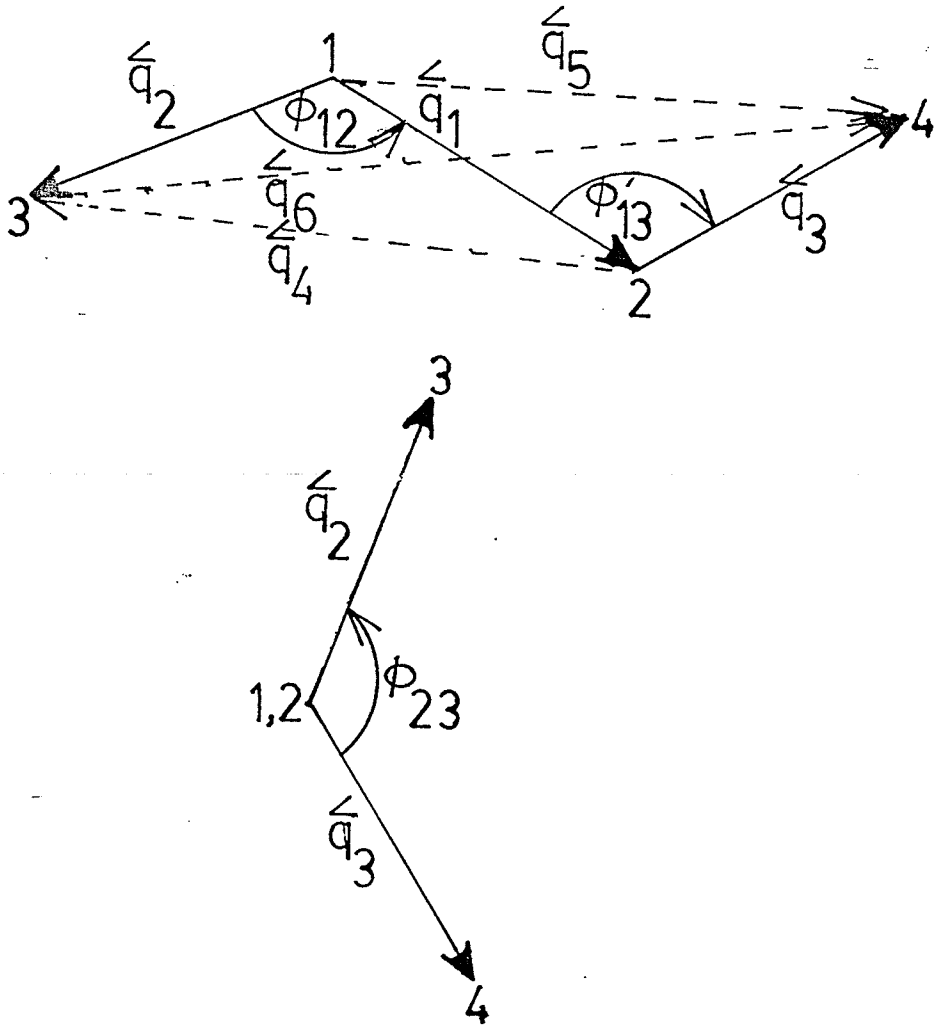

1
$\vec{q}_5$
$\vec{q}_2$
Φ_{12}
$\vec{q}_1$
4
3
$\vec{q}_6$
Φ'_{13}
$\vec{q}_3$
$\vec{q}_4$
2
3
$\vec{q}_2$
1,2
Φ_{23}
$\vec{q}_3$
4

FIG. 3.3

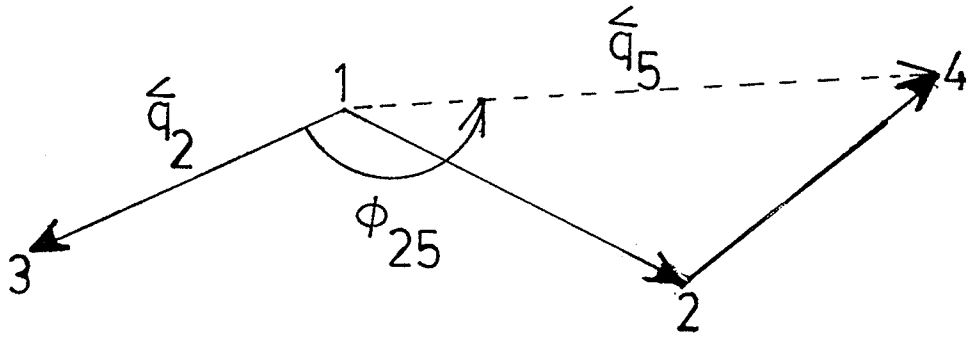

$\vec{q}_5$
1
4
$\vec{q}_2$
Φ_{25}
3
2

FIG. 3.4

$$\cos\Phi_{23} = (q_5\cos\Phi_{25} - q_1\cos\Phi_{12})/q_3 \cdot \qquad (3.12)$$

Similarly to Eqns. 3.9 to 3.11, $\cos\Phi_{25}$ is written as

$$\cos\Phi_{25} = \left[\frac{(q_2)^2 + (q_5)^2 - (q_6)^2}{2q_2q_5}\right] \cdot \qquad (3.13)$$

Likewise,

$$\cos\Phi_{12} = \left[\frac{(q_1)^2 + (q_2)^2 - (q_4)^2}{2q_1q_2}\right] \cdot \qquad (3.14)$$

Substitution of Eqns. 3.13 and 3.14 into Eqn. 3.12, yields Eqn. 3.11. The other three bond distances can be recovered by using Eqns. 3.15 to 3.17.

$$q_5 = [(q_1)^2 + (q_3)^2 - 2(q_1)(q_3)\cos(\pi-\Phi_{23})]^{1/2} \qquad (3.15)$$

$$q_6 = [(q_5)^2 + (q_4)^2 - (q_1)^2 - 2(q_2)(q_3)\cos\Phi_{23}]^{1/2} \qquad (3.16)$$

$$q_4 = [(q_1)^2 + (q_2)^2 - 2(q_1)(q_2)\cos\Phi_{12}]^{1/2} \qquad (3.17)$$

3.2.2.3. Restrictions On Bond Angles

Each time that the set Φ_{12}, Φ_{13}, and Φ_{23} is calculated, inequalities 3.18 to 3.21 must be checked [13] to ensure that the values of the bond distances and angles correspond to an actual physical configuration of the molecule. As well, the same conditions must be met by the Basic Rotational Invariant (BRI) [23] angles (see section 3.3.1.). The first condition (Eqn. 3.18) ensures that the sum of the angles is less than or equal to 2π, and the last three conditions (Eqns. 3.19 to 3.21) ensure that the sum of any two angles is greater than or equal to the third.

$$\Phi_{12} + \Phi_{13} + \Phi_{23} \leq 2\pi \tag{3.18}$$

$$\Phi_{12} + \Phi_{23} \geq \Phi_{13} \tag{3.19}$$

$$\Phi_{12} + \Phi_{13} \geq \Phi_{23} \tag{3.20}$$

$$\Phi_{13} + \Phi_{23} \geq \Phi_{12} \tag{3.21}$$

3.2.2.4. Calculation Of φ For the CHA Frame [16]

Due to coupling terms in the Hamiltonian of four atom molecules in the Curtiss Hirschfelder and Adler frame of reference ten, the angle φ is used instead of Θ_{23}. This is due to the fact that it is more reasonable to assume that the φ dependence of the potential can be partitioned into the nonseparable potential, than can the Θ_{23} dependence. The phi part of the Hamiltonian is then integrated out.

The three Generalized Jacobi Vectors are drawn in Fig. 3.5. In what

FIGURE 3.5: GJV'S OF FOUR BODY SHOWING ANGLE φ (CHA FRAME)

$\vec{Q}_i$ are the GJV's.

Θ_{ij} are the angles between the vectors.

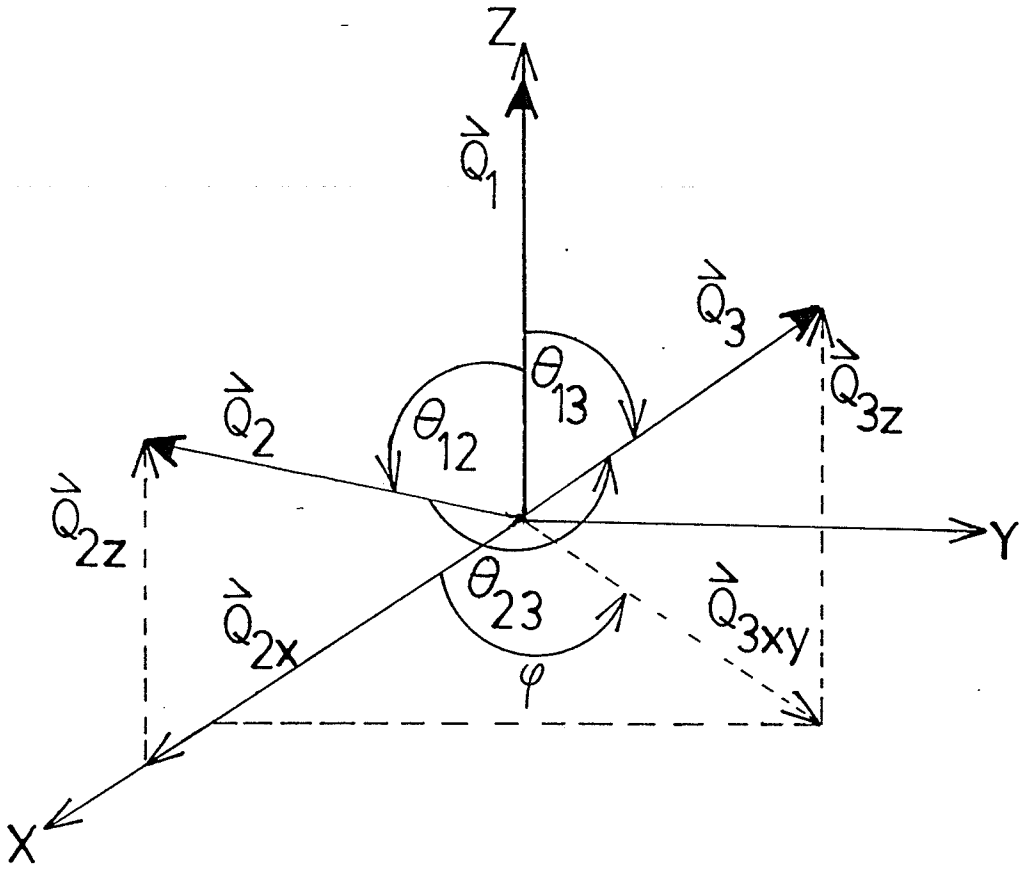

Z
$\vec{Q}_1$
$\vec{Q}_3$
$\vec{Q}_{3z}$
θ_{13}
$\vec{Q}_2$
θ_{12}
$\vec{Q}_{2z}$
Y
θ_{23}
$\vec{Q}_{3xy}$
$\vec{Q}_{2x}$
φ
X

FIG. 3.5

follows, Q_i denotes the magnitude of the Jacobi Vector $\vec{Q}_i$. The dot product of $\vec{Q}_2$ and $\vec{Q}_3$ yields

$$<\vec{Q}_2,\vec{Q}_3> = Q_2Q_3\cos\Theta_{23}. \tag{3.22}$$

$\vec{Q}_2$ and $\vec{Q}_3$ are written in terms of their components as

$$\vec{Q}_2 = \vec{Q}_{2x} + \vec{Q}_{2z} \tag{3.23}$$

$$\vec{Q}_3 = \vec{Q}_{3xy} + \vec{Q}_{3z} . \tag{3.24}$$

The dot product of Eqn. 3.22 can now be written as (using Eqns. 3.23 and 3.24)

$$<\vec{Q}_2,\vec{Q}_3> = Q_{2x}Q_{3xy}\cos\varphi + Q_{2z}Q_{3z} . \tag{3.25}$$

From Fig. 3.5 it can be seen that

$$Q_{2x} = Q_2\cos(\pi/2 - \Theta_{12}) \tag{3.26}$$

$$Q_{3xy} = Q_3\cos(\pi/2 - \Theta_{13}) \tag{3.27}$$

$$Q_{2z} = Q_2\sin(\pi/2 - \Theta_{12}) \tag{3.28}$$

$$\text{and} \quad Q_{3z} = Q_3\sin(\pi/2 - \Theta_{13}) . \tag{3.29}$$

Substitution of Eqns. 3.25 to 3.29 into Eqn. 3.22 gives

$$\cos\Theta_{23} = \sin\Theta_{12}\sin\Theta_{13}\cos\varphi + \cos\Theta_{12}\cos\Theta_{13} , \tag{3.30}$$

which, upon solving for $\cos\varphi$, gives

$$\cos\varphi = (\cos\Theta_{23} - \cos\Theta_{12}\cos\Theta_{13})/(\sin\Theta_{12}\sin\Theta_{13}) \ . \qquad (3.31)$$

Θ_{23} is recovered from Eqn. 3.30, as

$$\cos\Theta_{23} = \sin\Theta_{12}\sin\Theta_{23}\cos\varphi + \cos\Theta_{12}\cos\Theta_{13} \ . \qquad (3.32)$$

3.3. COORDINATE TRANSFORMATIONS

3.3.1. Introduction

The O matrices calculated by *GJVGEN* are used to transform the GRAM matrix of the bond distance–bond angle coordinates to the GRAM matrix of the BRI coordinates. This can be done since the contravariant base vectors (basis of the Metric tensor) and the physical 'bond' vectors of the molecule transform in an identical manner [10]. Once the BRI coordinates of the molecule are obtained, further transformations can be performed (see section 3.3.3.).

3.3.2. Basic Rotational Invariant

As indicated in section 3.3.1., the transformation of the Gram matrix by the orthonormalizing O matrix gives the Gram matrix of the BRI coordinates. An outline of the procedure used to obtain the BRI coordinates follows.

$q_1, q_2, \cdots, q_{3N-6}$ (Scalar Bond Distances)

$\Big\downarrow$ Form Matrix from $q_{i's}$ and inter bond angles.

$$\begin{bmatrix} q_1 & \Phi_{12} & \cdots & \Phi_{1(N-1)} \\ \Phi_{12} & q_2 & \cdots & \Phi_{2(N-1)} \\ \cdots & & & \\ \Phi_{(N-1)1} & \Phi_{(N-1)2} & \cdots & q_{N-1} \end{bmatrix}$$

$\Big\downarrow$ Generate Gram matrix.

$$G = \begin{bmatrix} \langle q_1,q_1\rangle & \langle q_1,q_2\rangle & \cdots & \langle q_1,q_{N-1}\rangle \\ \langle q_2,q_1\rangle & \langle q_2,q_2\rangle & \cdots & \langle q_2,q_{N-1}\rangle \\ \cdots & & & \\ \langle q_{N-1},q_1\rangle & \langle q_{N-1},q_2\rangle & \cdots & \langle q_{N-1},q_{N-1}\rangle \end{bmatrix} \quad (3.35)$$

G is the Gram matrix of the bond vectors. The transformation of the Gram matrix is

$$OGO^t = G' \tag{3.36}$$

where G' is the Gram matrix of the BRI coordinates, and

$$G' = \begin{bmatrix} \langle Q_1, Q_1 \rangle & \langle Q_1, Q_2 \rangle & \cdots & \langle Q_1, Q_{N-1} \rangle \\ \langle Q_2, Q_1 \rangle & \langle Q_2, Q_2 \rangle & \cdots & \langle Q_2, Q_{N-1} \rangle \\ \cdots & & & \\ \langle Q_{(N-1)}, Q_1 \rangle & \langle Q_{(N-1)}, Q_2 \rangle & \cdots & \langle Q_{(N-1)}, Q_{(N-1)} \rangle \end{bmatrix} . \tag{3.37}$$

Note that **any** O matrix obtained in *GJVGEN* (for the appropriate molecule) can be used here. From G', the magnitude of and angles between the Jacobi vectors, Q_i, can be obtained and stored in matrix form:

$$\downarrow$$

$$\begin{bmatrix} Q_1 & \theta_{12} & \cdots & \theta_{1(N-1)} \\ \theta_{21} & Q_2 & \cdots & \theta_{2(N-1)} \\ \cdots & & & \\ \theta_{(N-1)1} & \theta_{(N-1)2} & \cdots & Q_{N-1} \end{bmatrix} .$$

This final matrix contains N(N–1)/2 BRI coordinates. The range of these coordinates is $Q_i \geq 0$, and $-\infty < \Theta_{12} < \infty$.

3.3.3. Further Coordinate Transformations for Three Atom Molecules

3.3.3.1. Polar Coordinates

Polar coordinates are defined from the BRI coordinates defined in 3.3.2.. The transformation to polar coordinates (ρ, α, Θ_{12}) is given in Eqns. 3.38 to 3.40 [23].

$$\rho = [Q_1^{\,2} + Q_2^{\,2}]^{1/2} \tag{3.38}$$

$$\alpha = \arccos(Q_1/\rho) \;=\; \arcsin(Q_2/\rho) \tag{3.39}$$

$$\Theta_{12} = \Theta_{12} \tag{3.40}$$

The ranges of the polar coordinates are $\rho > 0$, $0 < \alpha < \pi/2$, and $-\infty < \Theta_{12} < \infty$. The transformation back to the BRI coordinates is

$$Q_1 = \rho\cos\alpha\,, \qquad Q_2 = \rho\sin\alpha\,, \text{ and} \tag{3.41}$$

$$\Theta_{12} = \Theta_{12}\,. \tag{3.42}$$

3.3.3.2. Johnson Hyperspherical

The transformation to Johnson hyperspherical coordinates (ρ,ϕ,ϕ) [16] is given by

$$\rho = (Q_1^{\,2} + Q_2^{\,2})^{1/2} , \tag{3.43}$$

$$\phi = (1/2)\tan^{-1}\left[\frac{\pm[Q_1^{\,4} + Q_2^{\,4} + 2Q_1^{\,2}Q_2^{\,2}(2\cos^2\Theta_{12} - 1)]^{1/2}}{2Q_1Q_2\sin\Theta_{12}}\right], \tag{3.44}$$

$$\phi = (1/2)\tan^{-1}\left[\frac{Q_1^{\,2} - Q_2^{\,2}}{2Q_1Q_2\cos\Theta_{12}}\right] , \tag{3.45}$$

The ranges of the hyperspherical coordinates are $\rho > 0$, $0 < \phi \leq \pi/4$, and $-\pi/4 \leq \phi \leq \pi/4$. The transformation back to the BRI coordinates is

$$Q_1 = \rho[1/2 + \sin2\phi\sin2\phi/2]^{1/2} , \tag{3.46}$$

$$Q_2 = \rho[1/2 - \sin2\phi\sin2\phi/2]^{1/2} , \text{ and} \tag{3.47}$$

$$\Theta_{12} = \sin^{-1}[\rho^2\cos2\phi/2Q_1Q_2] . \tag{3.48}$$

3.3.3.3. Standard (Smith) Hyperspherical

The transformation to standard hyperspherical coordinates (ρ,ϕ,ϕ) [16] is given by

$$\rho = (Q_1^2 + Q_2^2)^{1/2} , \tag{3.49}$$

$$\phi = (1/2)\tan^{-1}\left[\frac{2Q_1Q_2\sin\Theta}{\pm[Q_1^4 + Q_2^4 + 2Q_1^2Q_2^2(2\cos^2\Theta_{12} - 1)]^{1/2}} \right], \tag{3.50}$$

$$\text{and} \qquad \phi = (1/2)\tan^{-1}\left[\frac{2Q_1Q_2\cos\Theta_{12}}{Q_1^2 - Q_2^2} \right] . \tag{3.51}$$

The ranges of the hyperspherical coordinates are $\rho > 0$, $0 \leq \phi \leq \pi/4$, and $-\pi/4 \leq \phi \leq \pi/4$. The transformation back to the BRI coordinates is

$$Q_1 = \rho[1/2 + \cos2\phi\cos2\phi/2]^{1/2} , \tag{3.52}$$

$$Q_2 = \rho[1/2 - \cos2\phi\cos2\phi/2]^{1/2} , \text{ and} \tag{3.53}$$

$$\Theta_{12} = \sin^{-1}[\rho^2\sin2\phi/2Q_1Q_2] . \tag{3.54}$$

3.4. FORM OF ZERO–ORDER HAMILTONIAN

The form of the rotational ground state zero order Hamiltonian affects the form of the separable potentials (see section 3.5.) The zero–order Hamiltonians of the sets of coordinates of section 3.3. are given in sections 3.4.1. and 3.4.2. [4,10,19]. Notice that in each of the coordinate systems, the Hamiltonian is of the form of three one dimensional equations. Each of these equations is numerically soluble by the program

discussed in chapter 4.

3.4.1. Three Atom Molecules

3.4.1.1. Basic Rotational Invariant Coordinates

In what follows, the equilibrium value of any coordinate is written as $\bar{Q}_i$. For the BRI, the Hamiltonian is [10,14]

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_1^{\,2}} + \frac{2}{Q_1}\frac{\partial}{\partial Q_1}\right] + V(Q_1, \bar{Q}_2, \bar{\Theta}_{12}) \ , \tag{3.55}$$

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_2^{\,2}} + \frac{2}{Q_2}\frac{\partial}{\partial Q_2}\right] + V(\bar{Q}_1, Q_2, \bar{\Theta}_{12}) \ , \text{ and} \tag{3.56}$$

$$-(\hbar^2/2)\,(Q_1^{\,-2}+Q_2^{\,-2})\left[\frac{\partial^2}{\partial\Theta_{12}^{\,2}} + \cot\Theta_{12}\frac{\partial}{\partial\Theta_{12}}\right] + \frac{\hbar^2}{2}(Q_1^{-2}+Q_2^{-2})V'_{sep}(\Theta_{12}), \tag{3.57}$$

where $V'_{sep}(\Theta_{12}) = \frac{2}{\hbar^2}(\bar{Q}_1^{\,-2} + \bar{Q}_2^{\,-2})^{-1} V(\bar{Q}_1, \bar{Q}_2, \Theta_{12})$.

3.4.1.2. Polar Coordinates

The zero–order Hamiltonian is

$$-\frac{\hbar^2}{2}\left[\frac{\partial^2}{\partial\rho^2} - \frac{5}{\rho}\frac{\partial}{\partial\rho}\right] + V(\rho,\bar{\alpha},\bar{\Theta}_{12}) \ , \tag{3.58}$$

$$\frac{-\hbar^2}{2\rho^2}\left[\frac{\partial^2}{\partial\alpha^2} - 4\cot(2\alpha)\ \frac{\partial}{\partial\alpha}\right] + \frac{\hbar^2}{2\rho^2}V'_{sep}(\alpha) \ , \text{ and} \tag{3.59}$$

$$-\frac{\hbar^2}{2}\left[\frac{4\csc^2(2\alpha)}{\rho^2}\right]\left[\frac{\partial^2}{\partial\Theta_{12}^2} + \cot\Theta_{12}\ \frac{\partial}{\partial\Theta_{12}}\right] + \frac{\hbar^2}{2}\left[\frac{4\csc^2(2\alpha)}{\rho^2}\right]V'_{sep}(\Theta_{12}) \ , \tag{3.60}$$

where

$$V'_{sep}(\alpha) = \frac{2\bar{\rho}^2}{\hbar^2}V(\bar{\rho},\alpha,\bar{\Theta}_{12}) \ , \text{ and} \tag{3.61}$$

$$V'_{sep}(\Theta_{12}) = \frac{2}{\hbar^2}\left[\frac{\bar{\rho}^2}{4\csc^2(2\bar{\alpha})}\right]V(\bar{\rho},\bar{\alpha},\Theta_{12}) \ . \tag{3.62}$$

3.4.1.3. Johnson Hyperspherical Coordinates

The zero–order Hamiltonian is

$$-\frac{\hbar^2}{2}\left[\frac{\partial^2}{\partial\rho^2} + \frac{5}{\rho}\frac{\partial}{\partial\rho}\right] + V(\rho,\bar{\phi},\bar{\Phi}) \ , \tag{3.63}$$

$$\frac{-\hbar^2 2}{\rho^2}\left[\frac{\partial^2}{\partial\phi^2} + \cot\phi\frac{\partial}{\partial\phi} + \frac{1}{4\cos^2\phi}\right] + \frac{\hbar^2 2}{\rho^2}V'_{sep}(\phi) \ , \tag{3.64}$$

and

$$-\frac{\hbar^2}{2}\left[\frac{4}{\rho^2\sin^2\phi}\right]\left[\frac{\partial^2}{\partial\Phi^2}\right] + \frac{\hbar^2}{2}\left[\frac{4}{\rho^2\sin^2\phi}\right]V'_{sep}(\phi) \ , \tag{3.65}$$

where

$$V'_{sep}(\phi) = \frac{\bar{\rho}^2}{2\hbar^2}V(\bar{\rho},\phi,\bar{\Phi}) \ , \text{ and} \tag{3.66}$$

$$V'_{sep}(\Phi) = \frac{2}{\hbar^2}\left[\frac{\bar{\rho}^2\sin^2\bar{\phi}}{4}\right]V(\bar{\rho},\bar{\phi},\Phi) \ . \tag{3.67}$$

3.4.1.4. Standard Hyperspherical Coordinates

The zero–order Hamiltonian is

$$-\frac{\hbar^2}{2}\left[\frac{\partial^2}{\partial\rho^2} + \frac{5}{\rho}\frac{\partial}{\partial\rho}\right] + V(\rho,\bar{\phi},\bar{\Phi}) \ , \tag{3.68}$$

$$\frac{-\hbar^2}{2\rho^2}\left[\frac{\partial^2}{\partial\phi^2} + 4\cot(4\phi)\frac{\partial}{\partial\phi}\right] + \frac{\hbar^2}{2\rho^2}V'_{sep}(\phi) \ , \tag{3.69}$$

and $$-\frac{\hbar^2}{2}\left[\frac{64\cot^2(4\phi)}{\rho^2}\right]\left[\frac{\partial^2}{\partial\phi^2}\right] + \frac{\hbar^2}{2}\left[\frac{64\cot^2(4\phi)}{\rho^2}\right]V'_{sep}(\phi) \quad , \tag{3.70}$$

where

$$V'_{sep}(\phi) = \frac{2}{\hbar^2}\bar{\rho}^2 V(\bar{\rho},\phi,\bar{\Phi}) \text{ , and} \tag{3.71}$$

$$V'_{sep}(\phi) = \frac{2}{\hbar^2}\left[\frac{\bar{\rho}^2}{64\cot^2(4\bar{\phi})}\right]V(\bar{\rho},\bar{\phi},\phi) \quad . \tag{3.72}$$

3.4.2. Four Atom Molecules

3.4.2.1. Basic Rotational Invariants (CHA Frame) [14]

The zero–order Hamiltonian of the Curtiss Hirschfelder and Adler frame of reference three is

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_1^2} + \frac{2}{Q_1}\frac{\partial}{\partial Q_1} + 2Q_1^{-2}m^2\right] + V(Q_1,\bar{Q}_2,\bar{Q}_3,\bar{\Theta}_{12},\bar{\Theta}_{13}) \quad , \tag{3.73}$$

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_2^2} + \frac{2}{Q_2}\frac{\partial}{\partial Q_2}\right] + V(\bar{Q}_1,Q_2,\bar{Q}_3,\bar{\Theta}_{12},\bar{\Theta}_{13}) \quad , \tag{3.74}$$

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_3^{\,2}} + \frac{2}{Q_3}\frac{\partial}{\partial Q_3}\right] + V(\bar{Q}_1,\bar{Q}_2,Q_3,\bar{\Theta}_{12},\bar{\Theta}_{13}) \ , \qquad (3.75)$$

$$-(\hbar^2/2)(Q_1^{-2}+Q_2^{-2})\left[\frac{\partial^2}{\partial\Theta_{12}^{\,2}} + \cot\Theta_{12}\frac{\partial}{\partial\Theta_{12}} - m^2\csc^2\Theta_{12}\right] +$$

$$\frac{\hbar^2}{2}(Q_1^{-2}+Q_2^{-2})V'_{sep}(\Theta_{12}) \qquad (3.76)$$

and

$$-(\hbar^2/2)(Q_1^{-2}+Q_3^{-2})\left[\frac{\partial^2}{\partial\Theta_{13}^{\,2}} + \cot\Theta_{13}\frac{\partial}{\partial\Theta_{13}} - m^2\csc^2\Theta_{13}\right] +$$

$$\frac{\hbar^2}{2}(Q_1^{-2}+Q_3^{-2})V'_{sep}(\Theta_{13}) \qquad (3.77)$$

where $V'_{sep}(\Theta_{12}) = \frac{2}{\hbar^2}(\bar{Q}_1^{-2} + \bar{Q}_2^{-2})^{-1}V(\bar{Q}_1,\bar{Q}_2,\bar{Q}_3,\Theta_{12},\bar{\Theta}_{13})$,

and $V'_{sep}(\Theta_{13}) = \frac{2}{\hbar^2}(\bar{Q}_1^{-2} + \bar{Q}_3^{-2})^{-1}V(\bar{Q}_1,\bar{Q}_2,\bar{Q}_3,\bar{\Theta}_{12},\Theta_{13})$.

Note that (section 3.2.2.4.) the φ coordinate has been integrated out, yielding the quantum number m and eigenfunction $(2\pi)^{-1/2}\exp(im\varphi)$.

3.4.2.2. Basic Rotational Invariants (3GJV Frame) [10]

The zero–order Hamiltonian of the three Generalized Jacobi Vectors frame of reference four is

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_1^{\,2}} + \frac{2}{Q_1}\frac{\partial}{\partial Q_1}\right] + V(Q_1,\bar{Q}_2,\bar{Q}_3,\bar{\Theta}_{12},\bar{\Theta}_{13},\bar{\Theta}_{23}) \ , \qquad (3.78)$$

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_2^{\,2}} + \frac{2}{Q_2}\frac{\partial}{\partial Q_2}\right] + V(\bar{Q}_1,Q_2,\bar{Q}_3,\bar{\Theta}_{12},\bar{\Theta}_{13},\bar{\Theta}_{23}) \ , \qquad (3.79)$$

$$-(\hbar^2/2)\left[\frac{\partial^2}{\partial Q_3^{\,2}} + \frac{2}{Q_3}\frac{\partial}{\partial Q_3}\right] + V(\bar{Q}_1,\bar{Q}_2,Q_3,\bar{\Theta}_{12},\bar{\Theta}_{13},\bar{\Theta}_{23}) \ , \qquad (3.80)$$

$$-(\hbar^2/2)(Q_1^{-2}+Q_2^{-2})\left[\frac{\partial^2}{\partial\Theta_{12}^{\,2}}+\cot\Theta_{12}\frac{\partial}{\partial\Theta_{12}}\right]+\frac{\hbar^2}{2}(Q_1^{-2}+Q_2^{-2})V'_{sep}(\Theta_{12}) \ , \qquad (3.81)$$

$$-(\hbar^2/2)(Q_1^{-2}+Q_3^{-2})\left[\frac{\partial^2}{\partial\Theta_{13}^{\,2}}+\cot\Theta_{13}\frac{\partial}{\partial\Theta_{13}}\right]+\frac{\hbar^2}{2}(Q_1^{-2}+Q_3^{-2})V'_{sep}(\Theta_{13}) \ , \qquad (3.82)$$

and

$$-(\hbar^2/2)(Q_2^{-2}+Q_3^{-2})\left[\frac{\partial^2}{\partial\Theta_{23}^{\,2}}+\cot\Theta_{23}\frac{\partial}{\partial\Theta_{23}}\right]+\frac{\hbar^2}{2}(Q_2^{-2}+Q_3^{-2})V'_{sep}(\Theta_{23}) \ , \qquad (3.83)$$

where $V'_{sep}(\Theta_{12}) = \frac{2}{\hbar^2}(\bar{Q}_1^{-2} + \bar{Q}_2^{-2})^{-1}V(\bar{Q}_1,\bar{Q}_2,\bar{Q}_3,\Theta_{12},\bar{\Theta}_{13},\bar{\Theta}_{23})$,

$V'_{sep}(\Theta_{13}) = \frac{2}{\hbar^2}(\bar{Q}_1^{-2} + \bar{Q}_3^{-2})^{-1}V(\bar{Q}_1,\bar{Q}_2,\bar{Q}_3,\bar{\Theta}_{12},\Theta_{13},\bar{\Theta}_{23})$,

and $V'_{sep}(\Theta_{23}) = \frac{2}{\hbar^2}(\bar{Q}_2^{-2} + \bar{Q}_3^{-2})^{-1}V(\bar{Q}_1,\bar{Q}_2,\bar{Q}_3,\bar{\Theta}_{12},\bar{\Theta}_{13},\Theta_{23})$.

3.5. FORM OF POTENTIALS

The factor in front of the various terms of the above Hamiltonians determines the form of the potential. For example, in the case of the BRI of the tri–atoms, the separable potential of the third term is

$$(Q_1^{-2} + Q_2^{-2})(\bar{Q}_1{}^{-2} + \bar{Q}_2{}^{-2})^{-1} V(\bar{Q}_1, \bar{Q}_2, \Theta_{12}) \qquad (3.84)$$

The second term in Eqn. 3.84 ensures that the potential units are consistent with the units of the potentials of the first two parts of the Hamiltonian.

3.6. MOLECULAR POTENTIAL ENERGY FUNCTIONS

For an explanation of the construction of the potential functions employed in *NPLOT*, the reader is referred to the book **MOLECULAR POTENTIAL ENERGY FUNCTIONS** by MURRELL et al [15].

3.6.1. Three Atom Molecules

Each potential [15,17–19] is made up of three single atom terms, three diatomic terms, and a triatomic term.

$$V_{ABC} = \Sigma_A V_A^{(1)} + \Sigma_{AB} V_{AB}^{(2)} + V_{ABC} \qquad (3.85)$$

In most cases, the single atom terms are zero, since the molecule dissociates to ground state atoms [20].

The diatomic potentials are of the form

$$V^{(2)} = -D_e(1 + a_1\rho + a_2\rho^2 + a_3\rho^3)e^{(-a_1\rho)} \qquad (3.86)$$

where a_i are constants, $\rho = R - R_e$, and D_e is the dissociation energy. The Re's are also constants, not to be confused with the equilibrium bond distances of the molecule.

The triatomic part of the potential is of the form

$$V_{ABC} = V^0(1 + \sum_i C_i\rho_i + \sum_i \sum_{j \le i} C_{ij}\rho_i\rho_j + ...)[\prod_{i=1,3}(1-\tanh\gamma_i\rho_i/2)] \qquad (3.87)$$

T is a range function and equals 1 at equilibrium (i.e. $\rho = 0$).
The range function in some cases is $\Pi(1-\tanh\gamma_i S_i/2)$, where the S_i are symmetry combinations of the ρ_i [15].

3.6.2. Four Atom Molecules

The four atom potentials [15] are made up in an identical manner to the three atom potentials. In this case, though, there are four triatomic terms and an additional four body term:

$$V_{ABCD} = V^0(1 + \sum_i C_i\rho_i + \sum_i \sum_{j \le i} C_{ij}\rho_i\rho_j + \ldots)[\prod_{i=1,4} (1-\tanh\gamma_i\rho_i/2)]. \quad (3.88)$$

3.7. DRAW SUBROUTINE

3.7.1. Source of Subprogram

The plotting subroutine *DRAW* was adapted from a subprogram written by R. Wallace. Wallace's subprogram was derived from code developed by Dr. J. C. Eilbeck [22].

The first part of the plotting subroutine (lines 2 to 95) outputs the various titles passed from the subroutine *PLOT2*, and the second part of the plotting subroutine (lines 96 to 930) draws the contour lines of the potential energy function.

3.7.2. Plotting

The IBM Plotting System was used as a source of graphics subroutines. For sending output to an output device, *DRAW* uses the graphics subroutines of the plotting system. For example, to write 'title' to an output device, the subroutine PNOTES is called as

STATUS = PNOTES(1,0.0,97.0,5,'TITLE') .

The 1 indicates that the character string will be referenced from the lower left corner of the view area of the device. The 0.0 and 97.0 are the x and y coordinates of the string, respectively. The 5 indicates that the string is 5 characters long.

3.8. APPLICATIONS AND PROGRAM USAGE

The program *NPLOT* is applicable to three and four atom molecules. Before running the program, it is necessary to run *GJVGEN* to obtain an O matrix for the current molecule. For symmetrical three atom molecules, there are three different O matrices which can be generated by *GJVGEN*. Each of these O matrices can be used in *NPLOT* with any one of the four coordinate systems. The user therefore has the option of looking at twelve different sets of plots. For unsymmetrical three atoms, there are twenty different sets (five O matrices). For four atoms, there are many more options, due to the possibility of partitioning within *GJVGEN*.

To use *NPLOT*, a potential function for the molecule must be written (or amended from existing ones), compiled and linked with *NPLOT* and particular coordinate subroutines. Each potential subroutine is given the name *MOLPOT*, but different file names are used for disk storage.

Each coordinate set is also stored in a file which is given an appropriate name. In each coordinate file, there are six different subroutines named *TOJAC*, *FROMJA*, *SEPTAR*, *DIFF*, *QEQXEQ*, and *FNV*. In *TOJAC*, the particular coordinate transformation from the rotational invariants is carried out. In *FROMJA*, the opposite transformation is done. In *SEPTAR*, the separable potentials for the coordinate system are calculated. In *DIFF*, the non–separable potentials are calculated. *QEQXEQ* is called by the main program *NPLOT*, to calculate the equilibrium values of the target coordinates from the equilibrium values of the bond distance coordinates. This subroutine is identical in all but the four body CHA case. In this case, there is a further transformation made to calculate the angle φ. *FNV* calculates the values of the bond distance coordinates from the values of the target coordinates. It then calculates the value of the potential.

A small batch file has been written to allow the user to choose a particular molecular potential and coordinate system. Object codes for the potential and coordinate subroutines are stored on floppy disks. If the user wanted to run water and polar coordinates, the following would be typed in:

D:> RUN HOH POL <RETURN> .

The object codes for water and polar coordinates would then be copied from floppy disk to the virtual disk (D) and linking of all necessary files would begin. To run the program, the user merely types in *NPLOT* <Return>. The program is loaded into memory and execution begins.

Chapter 4

The Numerov Program

4.1. INTRODUCTION

Equations have been derived [10,14,24] for the zeroth order Hamiltonian for relative or 'vibrational' motion of N particles. These equations are of the form

$$[A(x)d^2/dx^2 + B(x)d/dx]\psi(x) = [V(x) - E]\psi(x) \qquad (4.1)$$

B(x) may be equal to zero. Programs for the numerical solution of Eqn. 4.1 had previously been written in Hewlett Packard Basic by J.P. Leroy and R. Wallace. These programs were adapted for a Microsoft Fortran 77 program developed on an IBM AT/PC. The program is simple to use and requires only the specification of A(x) and B(x), along with a minimum of additional input.

4.2. GENERAL ALGORITHM

The solution of 4.1 has been divided into two distinct algorithms; one is for the case $B(x) \neq 0$, the other for the case $B(x) = 0$. After $B(x)$ is specified (see section 4.7), the program automatically chooses the pertinent algorithm. In each case, a pair of two–term recursion relations [27] are employed to find a solution to the differential equation, starting at each bound:

$$R_n = U_n/Y_n - [Z_n/Y_n]R_{n-1}^{-1} \tag{4.2}$$

$$S_n = U_n/Z_n - [Y_n/Z_n]S_{n+1}^{-1} \tag{4.3}$$

where R_n, S_n, U_n, Y_n, and Z_n are functions defined in sections 3 and 4. R_n is calculated by forward iteration from the lower bound while S_n is obtained by backward iteration from the upper bound [26].

The error function, D(E), [26] evaluated at the matching grid point, m, (see section 4.6), is used to judge when the eigenvalue, E, has been computed to a preset degree of accuracy, where

$$D(E) = 1/S_{m+1} - R_m \, . \tag{4.4}$$

where m is the matching grid point of the two solutions.

4.3. ALGORITHM 1

This algorithm is used to solve those equations which do not contain a first derivative term, that is when $B(x) = 0$.

The terms R, S, U, Y, and Z are then defined as follows [11]:

$$Q_n = 1/A_n[V_n - E] \tag{4.5}$$

$$T_n = (h^2/12)Q_n \tag{4.6}$$

$$R_n = (1-T_{n+1})\psi_{n+1}/[(1-T_n)\psi_n] \tag{4.7}$$

$$S_n = (1-T_{n-1})\psi_{n-1}/[(1-T_n)\psi_n] \tag{4.8}$$

$$U_n = (2 + 10T_n)/(1-T_n) \tag{4.9}$$

$$Y_n = 1.0 \qquad Z_n = 1.0 \tag{4.10}$$

To demonstrate the use of the first algorithm, the anharmonic oscillator has been chosen. This oscillator has the following potential

$$V(x) = kx^2 + ax^3 + bx^4. \tag{4.11}$$

The differential equation is

$$d^2\psi(x)/dx^2 = [V(x) - E]\psi(x). \tag{4.12}$$

For small a and b values, the second order eigenvalues of this equation are given as [28]:

$$E_n = (2n + 1) - a^2A_n^{(2)} + bB_n^{(1)} - b^2B_n^{(2)}$$

where

$$A_n^{(2)} = 15/4(n^2 + n + 11/30)$$
$$B_n^{(1)} = 3/4(2n^2 + 2n + 1)$$
$$B_n^{(2)} = 1/8(34n^3 + 51n^2 + 59n + 21). \quad (4.13)$$

The first four eigenvalues are:

$E_0 = 1.0071$ $\quad$ $E_1 = 3.03455$

$E_2 = 5.087425$ $\quad$ $E_3 = 7.163175$

A comparison of the above values with Fig. 4.1 shows that good agreement is obtained at low values of n, but that the difference increases with increasing n. This is attributed to the fact that the second order energy aproximation has greater error for larger eigenvalues [28].

4.4. ALGORITHM 2

This algorithm is used to solve those equations which contain a first derivative term, that is, equations for which $B(x) \neq 0$.

In this case, the terms R, S, U, Y, and Z are defined as follows [27]

$$R_n = \psi_{n+1}/\psi_n \;\; ; \;\; S_n = \psi_{n-1}/\psi_n \quad (4.14)$$
$$Q_n = 1/A_n[V_n - E] \quad (4.15)$$
$$T_n = (h^2/12)Q_n \quad (4.16)$$
$$U_n = 2 + 12T_n \quad (4.17)$$
$$Y_n = 1 + 0.5h[B_n/A_n] \;\; ; \;\; Z_n = 1 - 0.5h[B_n/A_n] \quad (4.18)$$

FIGURE 4.1: COMPUTED EIGENVALUES FOR THE ANHARMONIC OSCILLATOR

```
**************************************************
***       THE ANHARMONIC  OSCILL.  EQUATION      ***
**************************************************

A(x) = 1.0                    B(x) = 0.0

            V(x) = X^2 + .01X^3 +.01X^4

 501 POINTS      XMIN = -10.0000     XMAX =  10.0000

   PSI(   -10.000) = 0.0    PSI(    10.000) = 0.0

  EIGENVALUE LIMITS:          .0000,          8.0000
```

NODES OF EIGENFUNCTIONS	EIGENVALUES
3.0	7.1768
2.0	5.0929
1.0	3.0361
.0	1.0073

FIG. 4.1

where h = (b–a)/(N–1), N is the number of points, b is the value of x at the upper bound, and a is the value of x at the lower bound. A_n, B_n, and V_n are the grid–point values of A(x), B(x), and V(x), respectively.

An example of an equation which has a first derivative term is the hydrogen atom radial equation [29]. This equation is

$$[(1/2)d^2/dr^2 + (1/r)d/dr]\psi(r) = [L(L+1)/2r^2 - 1/r - E]\psi(r) \qquad (4.19)$$

In atomic units, the eigenvalues are given by [29]

$$E_n = -1/2n^2 \qquad (4.20)$$

which gives for L = 0:

$$E_2 = -.125 \qquad E_3 = -0.055555 \qquad E_4 = -0.03125$$
$$E_5 = -0.02 \qquad E_6 = -0.0138$$

The eigenvalues calculated by the program are given in Fig. 4.2. The agreement with the values calculated by Eqn. 4.20 is good, except for E_6 which is in error by about ten percent.

FIGURE 4.2: COMPUTED EIGENVALUES FOR THE HYDROGEN ATOM RADIAL EQUATION (L=0)

```
**************************************************
***       THE HYDROGEN ATOM RADIAL EQUATION      ***
**************************************************

A(x) = 0.5                    B(x) = 1.0/X

              V(x) = -1/X+(S(S+1))/(2X^2)

3001 POINTS      XMIN =     .0000     XMAX =  70.0000

   PSI(       .000) = 0.0   PSI(     70.000) = 0.0

  EIGENVALUE LIMITS:         -.2000,          -.0070
```

NODES OF EIGENFUNCTIONS	EIGENVALUES
5.0	-.0124
4.0	-.0199
3.0	-.0312
2.0	-.0556
1.0	-.1250

FIG. 4.2

4.5. FROBENIUS SERIES EXPANSION AT THE BOUNDS

4.5.1. Introduction

In the case of Legendre type equations, a Frobenius series expansion at one or both bounds can be used [27].

For any equation

$$p(x)\psi''(x) + q(x)\psi'(x) + r(x)\psi(x) = 0 \qquad (4.21)$$

which has $p(a) = 0$, $x = a$ is a regular singular point if two conditions are fulfilled. These are i) $\lim (x-a)q(x)/p(x)$ exists as $x \to a$ and ii) $\lim (x-a)^2 r(x)/p(x)$ exists as $x \to a$. The Frobenius solution of Eqn. 4.21 is

$$\psi = (x-a)^c(\Sigma_j a_j(x-a)^j) \qquad (j = 0,1,2,...) \qquad (4.22)$$

where $\Sigma_j a_j(x-a)^j$ converges for all x such that $(x-a) < R$. R is the distance from $x = a$ to the nearest singularity.

The coefficients c and a_j are found by substituting ψ and its derivatives (calculated from Eqn. 4.22) into Eqn. 4.21. Terms of the same degree in x are set equal to 0.

4.5.2. Legendre–Type Equations

The Legendre equation with a potential V(x) can be written in the form

$$(1-x^2)\psi''(x) - 2x\psi'(x) = [s^2/(1-x^2) + V(x) - E]\psi(x) \qquad (4.23)$$

for $-1 < x < 1$. $x = -1$ is a regular singular point [32]. If V(x) can be written as

$$V(x) = \Sigma_i A_i(x-x_o)^i \quad (i = 1,2,...) \qquad (4.24)$$

where x_o is the boundary value of x, then the procedure outlined in the introduction can be used. It is found that

$$c = s/2$$

$$a_0 = 1$$

$$a_1 = (s^2 - s - 2E)/4(s + 1)$$

and

$$a_2 = [s^4+5s^3-4s^2(E-2)-4s(2E-2A_1-1)+4(E^2-2E+2A_1)]/[32(s+1)(s+2)] \; . \qquad (4.25)$$

Near $x = -1$, the solution is

$$\psi(x) = (x+1)^{s/2} + a_1(x+1)^{s/2+1} + a_2(x+1)^{s/2+2} \; . \qquad (4.26)$$

$R_1 = \psi(-1+h)/\psi(-1)$ can be evaluated by calculating $\psi(-1)$ and $\psi(-1+h)$ via Eqn. 4.26. As many values of ψ and R as desired can be calculated by using Eqn. 4.26. The upper bound solution is treated analogously, using an expansion in (x–1).

When a new trial eigenvalue is calculated by the procedures outlined in 4.6., it is used to calculate, consecutively, new values for the coefficients (Eqn. 4.25), new values for $\psi(x)$ (Eqn. 4.26), and new values for R_1, R_2, etc. The Frobenius expansion at the bounds differs from the other boundary conditions available in that the values of ψ at and near the bound depend on the current value of the eigenvalue.

4.6. ZEROING IN ON THE EIGENVALUE

E_n is the eigenvalue with a corresponding eigenfunction that has n nodes. Two different procedures are used to calculate the (i+1)th estimate of E_n from the ith value. The first procedure, which uses Newton's Method [30] is used initially in each case. This method proceeds as follows.

Successive aproximations to E_n are calculated from

$$E_n^{\,i+1} = E_n^{\,i} - T_1 D(E_n^i)/T_2$$

where,

$$T_1 = |E_n^i - E^{i-1}| \; ; \; T_2 = |D(E_n^i) - D(E_n^{i-1})|$$

and

$$D(E_n^i) = 1/S(m) - R(m) \ . \qquad (4.27)$$

The matching grid point, m, is determined as the first S (in backwards iteration from the upper bound) element which is less than or equal to one. The eigenfunction is at an extremum here. When two successive values of E^i are within a set amount of each other, iteration stops.

If the eigenvalue calculated was E_{n+1}, a new and smaller initial estimate is calculated, and iterations are resumed. The above procedure continues until E_n is calculated, or until an eigenvalue is calculated with a node count less than n. This eigenvalue is denoted by E_L (E_{Low}) [25]. Since the eigenvalues are calculated in descending order (each successively calculated eigenvalue is smaller than the previous one), E_{n+1} is known. Let E_{n+1} be denoted by E_H. The next estimate for E_n, E_{new}, is then $(E_L + E_H)/2$. If the node count of this new estimate is greater than n, E_H is set equal to E_{new}. If it is lower, E_L is set equal to E_{new}. If the node count is n, then it is not known whether the new estimate is greater than or less than the true value. A calculation of the second order energy correction, $(E_{correct} - E_{new})$, will give the needed additional information. If the sign of the correction is positive, let $E_L = E_{new}$. If it is negative, let $E_H = E_{new}$.

The sign of the second order energy correction is calculated as follows. A simple iteration formula can be worked out for the eigenvalue [31]:

$$E_{correct} - E_{new} = \frac{\psi(x_1)A(x_1)[d\psi/dx(x_1^+) - d\psi/dx(x_1^-)]}{\int_a^b \psi^* \psi dx} \qquad (4.28)$$

Only the sign of $E_{correct} - E_{new}$ is needed, and Eqn. 4.28 becomes

$$\text{sign}(\Delta E) = \text{sign}\{-A(x_1)[d\psi/dx(x_1^+) - d\psi/dx(x_1^-)]\}, \qquad (4.29)$$

since the integral, $\int_a^b$, is positive and $\psi(x_1)$ is set equal to positive one. $d\psi/dx(x_1^+)$ and $d\psi/dx(x_1^-)$ indicate the values of the eigenfunctions derivatives approached from the right (+,S_n is used) and left (–,R_n is used). They are calculated from [31]

$$hd\psi/dx = [1/2 - T_{n+1}]\psi_{n+1} - [1/2 - T_{n-1}]\psi_{n-1} \, . \qquad (4.30)$$

Depending on the algorithm, ψ_{n+1} and ψ_{n-1} are obtained from Eqns. 4.7 and 4.8 or Eqn. 4.14 with $\psi_n = 1.0$.

The bisection procedure is used for as many iterations as are needed to obtain $|E^{i+1} - E^i| < \epsilon$, where ϵ is a small preset constant. When this occurs, the Newton's method is used again, starting with E^{i+1} as its first estimate. This last switch in procedures is done since the Numerov method has been found to give more accurate results than the bisection method.

4.7. PROGRAM ORGANIZATION

A subroutine, *NUMROV*, has been written to be called by a user defined driver. The driver supplied prompts the user for the input parameters of NUMROV, which is then called. The driver may also call an output subroutine which prints the eigenvalues of the equation.

For each particular equation to be solved, two small subroutines must be written (or amended from existing ones), compiled, and linked with the rest of the program. The first of these subroutines calculates the two functions A(x) and B(x) and passes them back to the driver. The second subroutine calculates the potential V(x) and passes it back to the driver.

Chapter 5

Conclusions and Suggestions for Further Work

There is now available FORTRAN–77 code which enables the user, with a minimum of effort, to obtain a variety of matrices which are used to generate orthonormal G.J.V.'s for molecules with two to fourteen atoms.

A second program can be used to obtain contour plots of molecular potential energy functions for three and four atom molecules in four different coordinate systems.

A third program can solve the zero order Hamiltonians for various coordinate systems, once the form of the Hamiltonian, and an 'effective' potential for it are available. This generation of 'effective' potentials is the main programming stumbling block in the search for zero order eigenfunctions of the Hamiltonian.

The writing of code which yields the 'effective' potential for any user chosen range of a particular coordinate, is one possible area of future programming endeavors. This approach differs from that used earlier (indicated in the flow chart in the introduction). The previous procedure was to obtain a polynomial curve fit to the separable potential calculated in *NPLOT*. The problem with this method is that the polynomial curve fit can break down at the bounds. As well, in *NUMROV*, the user is now

restricted to use the upper and lower bound values for 'x' that were chosen in the curve fitting routine.

In the new approach, *VCALC* code could be written which is similar to the code which calculates the separable potential in *NPLOT*. It would differ from the code in *NPLOT*, in that any other terms in the Hamiltonian which contributed to the 'effective' potential would be included. In this way, the user would have more flexibility to choose the boundary values of 'x'.

References

1. R. Wallace, "Large–Amplitude Vibration in Triatoms II. Introduction of Potential Refinement Within a LAV Model Applied to the Sulfur Dioxide Molecule," Chem. Phys., vol. 74, pp. 339–345, February 1, 1983.

2. R. Wallace, "A Theory of Nuclear Motion in Polyatomic Molecules Based Upon the Morse Oscillator," Chem. Phys., vol. 11, pp. 189–199, 1975.

3. P. R. Stannard, M. L. Elert, and W. M. Gelbart, " On the Overtone–Combination Spectra of XY_2 Molecules[a)]," J. Chem. Phys., vol. 74, pp. 6050–6062, 1981.

4. R. Wallace, " The Selection of Coordinate Systems for Describing Large Amplitude Nuclear Motion in Triatomic Molecules," Chem. Phys., vol. 71, No. 2, pp. 173–180, October 1, 1982.

5. R. Wallace, "Large Amplitude Vibration in Polyatomic Molecules I. A Polar Representation of Orthogonal Relative Coordinates," Chem. Phys., vol. 88, No. 2, pp. 247–260, August 1984.

6. C. F. Curtiss, J. O. Hirschfelder, and F. T. Adler, "The Separation of the Rotational Coordinates from the N–Particle Schroedinger Equation," J. Chem. Phys., vol. 18, pp. 1638–1642, 1950.

7. J. P. Leroy and R. Wallace, "Form of the Quantum Kinetic Energy Operator for Relative Motion of a Group of Particles in a General Non–Inertial Reference Frame," Chem. Phys., vol. 118, pp. 379–395, 1987.

8. J. P. Leroy and R. Wallace, " The Quantum Kinetic Energy Operator for a Group of Particles in Terms of Scalar Basic Rotational Invariant Coordinates Derived from a Generalized Jacobi Vectors (GJV) Description II. Frames Derived from Three GJV," submitted to Chem. Phys., 1988.

9. J. P. Leroy and R. Wallace, " The Quantum Kinetic Energy Operator for a Group of Particles in Terms of Scalar Basic Rotational Invariant Coordinates Derived from a Generalized Jacobi Vectors (GJV) Description I. Frames Derived from Two GJV," to be published in Chem. Phys., 1988.12. K. R. Symon, Mechanics, Addison–Wesley Publishing Company, Reading, Massachusetts, 1971.

10. J. P. Leroy and R. Wallace, "Procedures Leading to a Variety of Orthogonal Jacobi–Type Coordinates of Relevance to Large–Amplitude Vibration and Scattering Problems," Chem. Phys., vol. 111, pp. 11–16, 1987.

11. J. P. Leroy and R. Wallace, "Renormalized Numerov Method Applied to Eigenvalue Equations: Extension to Include Single Derivative Terms and a Variety of Boundary Conditions," J. Phys. Chem., vol. 89, pp. 1928–1932, 1985.

12. K. R. Symon, *Mechanics*, Addison–Wesley Publishing Company, Reading, Massachusetts, 1971.

13. R. M. Thrall and L. Tornheim, *Vector Spaces and Matrices*, John Wiley and Sons, New York, 1957.

14. G. Bachman and L. Narici, *Functional Analysis*, Academic Press, New York, New York, 1966.

15. H. Anton, *Elementary Linear Algebra*, John Wiley & Sons, Inc., New York, New York, 1973.

16. J. P. Leroy and R. Wallace, "The Determination of Generalized Jacobi Vectors (GJV) for Common Types of Small Molecules," to be published in Chem. Phys., 1988.

17. GJVGEN manual submitted to Comput. Phys. Comms.

18. F. Ayres, Jr., *Trigonometry*, McGraw Hill, New York, New York, 1967.

19. B. R. Johnson, "On Hyperspherical Coordinates and Mapping the Internal Configurations of a Three Body System," J. Chem. Phys. vol. 73, No. 10, pp. 5051–5058, November 15, 1980.

20. J. P. Leroy, "The Quantum Kinetic Energy Operator for Arbitrary Motion of a Group of Particles in Terms of Generalized Jacobi Vectors and General Noninertial Frame," Ph.D. Dissertation, University of Manitoba, Winnipeg, Canada, 1988.

21. J. N. Murrell, S. Carter, and L. O. Halonen, "Frequency Optimized Potential Energy Functions for the Ground–State Surfaces of HCN and HCP," J. Mol. Spec., vol. 93, pp. 307–316, 1982.

22. J. N. Murrell, *Molecular Potential Energy Functions*, John Wiley and Sons, New York, New York, 1984.

23. K. S. Sorbie and J. N. Murrell, "Analytical Potentials for Triatomic Molecules from Spectroscopic Data," Mol. Phys., vol. 29, No. 5, pp. 1387–1407, 1975.

24. J. N. Murrell, W. Craven, M. Vincent, and Z. H. Zhu, "Potential Energy Functions for the Ground State Surfaces of SO_2 and S_2O," Mol. Phys., vol. 56, No. 4, pp. 839–851, 1985.

25. N. C. Handy and S. Carter, "An Improved Potential Surface for Formaldehyde," Chem. Phys. Lett., vol. 79, No.1, pp. 118–124, 1981; Errata, N. C. Handy and S. Carter, Chem. Phys. Lett., vol. 83, No. 1, p. 216.

26. Dr. Wallace personal communication.

27. J. P. Leroy and R. Wallace, " Extension of the Renormalized Numerov Method for Second–Order Differential Eigenvalue Equations," J. Comput. Phys., vol. 67, pp. 239–252, 1986.

28. S. Flügge, *Practical Quantum Mechanics*, Springer–Verlag, New York, New York, 1971.

29. I. N. Levine, *Quantum Chemistry*, Allyn and Bacon, Inc., Boston, 1970.

30. W. R. Derrick and S. I. Grossman, *Elementary Differential Equations with Applications*, Addison–Wesley Publishing Company, Reading, Massachusetts, 1981.

31. P. A. Collens et al, *Structured Fortran with WATFIV–S and WATFOR–11S*, Winnipeg, Manitoba, 1976.

32. B. R. Johnson, " New Numerical Methods Applied to Solving the One Dimensional Eigenvalue Problem[a.)]," J. Chem. Phys., vol. 67, No. 9, pp. 4086–4093, Nov. 1, 1977.

33. J. M. Blatt, " Practical Points Concerning the Solution of the Schrodinger Equation," J. Comput. Phys., vol. 1, pp. 382–396, 1967.

34. L. Halonen, M. S. Child, and S. Carter, " Potential Models and Local Mode Vibrational Eigenvalue Calculations for Acetylene," Mol. Phys., vol. 47, pp. 1097–1112, 1982.

Appendix A

Fortran–77 Source Code for *GJVGEN*

```
$NOFLOATCALLS
$STORAGE:2
C
C
C
      PROGRAM GJVGEN
C
C
      INTEGER*2 MAX20,MAX19,I,N,D,SECTIO,DUMMY
      REAL*8 M(14,14),WORK2(14,14),GRAM(14,14),GREL(13,13),CMNORM,DET,
     X       STOR2(13,13),SMOLEC(13,13),SCOMP(13,13),STOR1(13,13),
     X       GCMREL(14,14),WORK4(14,14),TCMREL(14,14),TOPMAT(14,14)
      CHARACTER MOLNAM*20,METHOD*1,GLOBAL*1,SNAME*20
C
C
      MAX20=14
      MAX19=13
5     CALL WRITEL(25)
      WRITE(*,*) 'GIVE THE NUMBER OF ATOMS IN THIS MOLECULE.'
      CALL WRITEL(10)
      READ(*,*) N
      IF ((N .GT. MAX20) .OR. (N .LT. 2)) THEN
        CALL WRITEL(3)
        WRITE(*,10) N
10      FORMAT(' THIS PROGRAM CAN''T HANDLE A ',I2,' BODY PROBLEM.')
        CALL WRITEL(10)
        CALL HOLD
```

```
      GOTO 5
    ENDIF
    CALL WRITEL(25)
    DO 20 I=1,N
      WRITE(*,15) I
15     FORMAT(' GIVE THE MASS OF ATOM ',I2,'.')
      WRITE(*,*)
      READ(*,*) M(I,I)
      CALL WRITEL(2)
20   CONTINUE
    CALL WRITEL(25)
    WRITE(*,*) 'GIVE THE NAME OF THIS MOLECULE.'
    CALL WRITEL(10)
    READ(*,25) MOLNAM
25   FORMAT(A20)
    CALL WRITEL(25)
    WRITE(*,*) 'ALL O MATRIX CALCULATIONS ARE DONE ON THE CONTRAVARIAN
   XT BASES.'
    CALL WRITEL(10)
    WRITE(*,*) 'PLEASE WAIT ...'
    CALL HOLD
    D=N
    CALL GRCALC(GRAM,M,D,MAX20,WORK2,TOPMAT,TCMREL,WORK4,GCMREL,GREL,
   X         CMNORM,MAX19)
    D=N-1
    CALL ASSIGN(STOR2,GREL,D,MAX19)
30   CALL WRITEL(25)
    WRITE(*,35) N-1
35   FORMAT(' DO YOU WANT TO PARTITION THE ',I2,' VECTOR(S) INTO GROUPS
   X, OR TREAT THEM GLOBALLY ?')
    WRITE(*,*) '(P/G)'
    CALL WRITEL(10)
    READ(*,40) METHOD
40   FORMAT(A1)
    IF ((METHOD .EQ. 'p') .OR. (METHOD .EQ. 'P')) THEN
      CALL PART(D,MAX19,MAX20,N,M,WORK2,SECTIO,GRAM,GREL,STOR2,
   X          SMOLEC,SCOMP,STOR1,GCMREL,WORK4,TCMREL,METHOD)
    ELSE
      IF ((METHOD .NE. 'G') .AND. (METHOD .NE. 'g')) THEN
        CALL ERRORS
        GOTO 30
      ENDIF
    ENDIF
125  GLOBAL='G'
    CALL SETUP(STOR1,D,MAX20,MAX19,GREL,M,GRAM,TCMREL,WORK2,WORK4,
   X         GCMREL,I,GLOBAL,DUMMY)
    CALL WRITEL(3)
    WRITE(*,*) 'GIVE A NAME DESCRIBING THE ORTHONORMALIZATION PROCEDUR
   XE(S) CHOSEN .'
    READ(*,126) SNAME
126  FORMAT(A20)
```

```
      IF ((METHOD .EQ. 'P') .OR. (METHOD .EQ. 'p')) THEN
        D=N-1
        CALL MATMUL(SMOLEC,STOR1,SCOMP,D,MAX19)
      ELSE
        CALL ASSIGN(SMOLEC,STOR1,D,MAX19)
      ENDIF
      CALL MATRAN(STOR1,SMOLEC,D,MAX19)
      CALL MATMUL(GREL,STOR2,STOR1,D,MAX19)
      CALL MATMUL(STOR1,SMOLEC,GREL,D,MAX19)
      CALL ASSIGN(GREL,STOR1,D,MAX19)
      CALL INVDET(STOR1,SMOLEC,D,MAX19,DET,STOR2)
      CALL OUTPUT(SNAME,SMOLEC,MOLNAM,STOR1,MAX19,D,SECTIO,METHOD,GREL)
      STOP
      END



C
C
C
      SUBROUTINE PART(D,MAX19,MAX20,N,M,WORK2,SECTIO,GRAM,GREL,STOR2,
     X            SMOLEC,SCOMP,STOR1,GCMREL,WORK4,TCMREL,METHOD)
C
C
      INTEGER*2 MAX20,MAX19,COUNT,COUNT3,COUNT4,I,K,L,N,D,BONDS,SECTIO,
     X       NSEC(9),GNUM,FIR,COUNT2,J
      REAL*8 M(MAX20,1),WORK2(MAX20,1),GRAM(MAX20,1),GREL(MAX19,1),AC,
     X     STOR2(MAX19,1),SMOLEC(MAX19,1),STOR3(9,9),SCOMP(MAX19,1),
     X     STOR1(MAX19,1),GCMREL(MAX20,1),WORK4(MAX20,1),
     X     TCMREL(MAX20,1),VSTOR(9),NUM
      CHARACTER METHOD*1,SECANS(9)*1,GLOBAL*1,ANS*1
      LOGICAL FIRST
C
C
      FIRST=.TRUE.
      NUM=0.0
      AC=1.0D-07
42    CALL WRITEL(25)
      WRITE(*,45) D
45    FORMAT(' HOW MANY GROUPS DO YOU WANT TO PARTITION THE ',I2,' VECTO
     XR(S) INTO ?')
      CALL WRITEL(10)
      READ(*,*) SECTIO
      IF ((SECTIO .GT. (N-1)) .OR. (SECTIO .LT. 1)) THEN
        CALL ERRORS
        GOTO 42
      ENDIF
50    BONDS=0
```

```
      DO 65 I=1,SECTIO
55      CALL WRITEL(25)
        WRITE(*,60) I
60       FORMAT(' HOW MANY VECTORS ARE IN GROUP ',I2,' ?')
        CALL WRITEL(10)
        READ(*,*) NSEC(I)

        IF ((NSEC(I) .GT. N-1) .OR. (NSEC(I) .LT. 1)) THEN
          CALL ERRORS
          GOTO 55
        ENDIF
        BONDS=BONDS+NSEC(I)
        NSEC(I)=NSEC(I)+1
65    CONTINUE
      IF (BONDS .NE. (N-1)) THEN
        CALL WRITEL(3)
        WRITE(*,70) BONDS
70       FORMAT(' ERROR !! THE TOTAL NUMBER OF VECTORS (',I2,') IS INCOR
     XRECT .')
        CALL WRITEL(10)
        CALL HOLD
        GOTO 50
      ENDIF
      DO 75 I=1,(N-1)
        SCOMP(I,I) = 1.0
75    CONTINUE
      COUNT=0
      DO 94 I=1,SECTIO
77      CALL WRITEL(25)
        WRITE(*,80) I
80       FORMAT(' DO YOU WANT TO ORTHONORMALIZE THE VECTOR(S) OF GROUP '
     X,I2,' ? (Y/N)')
        CALL WRITEL(10)
        READ(*,85) SECANS(I)
85       FORMAT(A1)
        IF ((SECANS(I) .EQ. 'Y') .OR. (SECANS(I) .EQ. 'y')) THEN
          DO 91 K=(COUNT+1),(COUNT+NSEC(I)-1)
            COUNT3=K-COUNT
            DO 90 L=(COUNT+1),(COUNT+NSEC(I)-1)
              COUNT4=L-COUNT
              GREL(COUNT3,COUNT4)=STOR2(K,L)
90           CONTINUE
91         CONTINUE
          D=NSEC(I)-1
          CALL SETUP(STOR3,D,MAX20,MAX19,GREL,M,GRAM,TCMREL,WORK2,
     X              WORK4,GCMREL,I,METHOD,GNUM)
          DO 93 K=(COUNT+1),(COUNT+NSEC(I)-1)
            COUNT3=K-COUNT
            DO 92 L=(COUNT+1),(COUNT+NSEC(I)-1)
              COUNT4=L-COUNT
              SCOMP(K,L)=STOR3(COUNT3,COUNT4)
92           CONTINUE
93         CONTINUE
```

```
      ELSE
        IF ((SECANS(I) .NE. 'N') .AND. (SECANS(I) .NE. 'n')) THEN
          CALL ERRORS
          GOTO 77
        ENDIF
      ENDIF
      COUNT=NSEC(I)-1+COUNT
94    CONTINUE
95    D=N-1
      CALL MATMUL(STOR1,SCOMP,STOR2,D,MAX19)
      CALL MATRAN(STOR3,SCOMP,D,MAX19)
      CALL MATMUL(GREL,STOR1,STOR3,D,MAX19)
      CALL WRITEL(25)
      WRITE(*,*) 'NEW METRIC TENSOR OF THE ENTIRE MOLECULE:'
      WRITE(*,*) '========================================='
      CALL WRITEL(2)
      DO 97 I=1,N-1
        WRITE(*,96) (GREL(I,J),J=1,N-1)
96      FORMAT(' ',20(F7.3,1X))
97    CONTINUE
      WRITE(*,*)
      IF (FIRST) THEN
        CALL WRITEL(5)
101     WRITE(*,*) 'DO YOU WANT TO ORTHONORMALIZE CERTAIN VECTORS ? (Y/
     XN)'
        READ(*,102) ANS
102     FORMAT(A1)
        IF ((ANS .NE. 'Y') .AND. (ANS .NE. 'y') .AND. (ANS .NE. 'N')
     X      .AND. (ANS .NE. 'n')) THEN
          CALL ERRORS
          GOTO 101
        ENDIF
        GOTO 105
      ENDIF
103   WRITE(*,*)
      WRITE(*,*) 'DO YOU WANT TO ORTHONORMALIZE ANY MORE VECTORS ? (Y/N)
     X'
      READ(*,104) ANS
104   FORMAT(A1)
105   IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
106     CALL MATSCA(STOR1,NUM,STOR3,D,MAX19)
        DO 108 K=1,N-1
          VSTOR(K)=0.0
108     CONTINUE
        WRITE(*,*)
        WRITE(*,*) 'HOW MANY VECTORS DO YOU WANT ORTHONORMALIZED ?'
        READ(*,*) GNUM
        IF ((GNUM .GT. N-1) .OR. (GNUM .LT. 1)) THEN
          CALL ERRORS
          GOTO 106
        ENDIF
        WRITE(*,*)
        WRITE(*,*) 'WHICH VECTORS ARE TO BE ORTHONORMALIZED ?'
```

```
      DO 110 K=1,GNUM
109      READ(*,*) FIR
        IF ((FIR .GT. N-1) .OR. (FIR .LT. 1) .OR.
     X(VSTOR(FIR) .EQ. 1.0)) THEN
          CALL ERRORS
          GOTO 109
        ENDIF
        VSTOR(FIR)=1.0
        STOR1(K,FIR)=1.0
110     CONTINUE



C
C       FINISH CALCULATING THE PERMUTATION MATRIX, P, AND STORE IN
C       STOR1.
C
      DO 112 K=(GNUM+1),N-1
        DO 111 J=1,N-1
          IF (VSTOR(J) .NE. 1.0) THEN
            VSTOR(J)=1.0
            STOR1(K,J)=1.0
            GOTO 112
          ENDIF
111       CONTINUE
112     CONTINUE
C
C       CALCULATE PGP^t AND STORE IN GREL. STORE P^t IN STOR3.
C
      CALL MATMUL(STOR3,STOR1,GREL,D,MAX19)
      CALL MATRAN(GREL,STOR1,D,MAX19)
      CALL MATMUL(STOR1,STOR3,GREL,D,MAX19)
      CALL ASSIGN(STOR3,GREL,D,MAX19)
      CALL ASSIGN(GREL,STOR1,D,MAX19)
C
C       CALCULATE S MATRIX FOR SUB-SPACE OF DIMENSION GNUM X GNUM.
C
      D=GNUM
      GLOBAL='C'
      CALL SETUP(STOR1,D,MAX20,MAX19,GREL,M,GRAM,TCMREL,WORK2,
     X          WORK4,GCMREL,I,GLOBAL,GNUM)
      DO 115 I=1,N-1
        DO 114 J=1,N-1
          IF (I .EQ. J) THEN
            GREL(I,I)=1.0
          ELSE
            GREL(I,J)=0.0
          ENDIF
114       CONTINUE
115     CONTINUE
      CALL ASSIGN(GREL,STOR1,D,MAX19)
      D=N-1
```

```
C
C     PERFORM INVERSE OF PERMUTATION TRANSFORMATION ON THE S MATRIX
C     FOR THE GNUM X GNUM DIMENSIONAL SUBSPACE.
C
      CALL MATMUL(STOR1,STOR3,GREL,D,MAX19)
      CALL MATRAN(GREL,STOR3,D,MAX19)
      CALL MATMUL(STOR3,STOR1,GREL,D,MAX19)
C
C     CALCULATE TOTAL TRANSFORMATION (S MATRIX) UP TO THIS POINT AND
C     ASSIGN TO SCOMP.
C
      CALL MATMUL(STOR1,STOR3,SCOMP,D,MAX19)
      CALL ASSIGN(SCOMP,STOR1,D,MAX19)
      FIRST=.FALSE.
      GOTO 95
    ELSE
      IF ((ANS .NE. 'N') .AND. (ANS .NE. 'n')) THEN
        CALL ERRORS
        GOTO 103
      ENDIF
    ENDIF
    D=N-1
    CALL MATMUL(STOR1,SCOMP,STOR2,D,MAX19)
    CALL MATRAN(STOR3,SCOMP,D,MAX19)
    CALL MATMUL(GREL,STOR1,STOR3,D,MAX19)
    COUNT=0
    DO 120 I=1,SECTIO
      IF ((SECANS(I) .EQ. 'Y') .OR. (SECANS(I) .EQ. 'y')) THEN
        DO 117 J=(COUNT+1),(COUNT+NSEC(I)-1)
          IF (ABS(GREL(J,J)-1.0) .GT. AC) THEN
            WRITE(*,*) 'ERROR !! INCORRECT ORDERING OF GROUPS, OR
   XINCORRECT MASS ASSIGNMENT WITHIN'
            WRITE(*,*) 'GROUP(S).'
            CALL WRITEL(2)
            WRITE(*,116) J,J,I,GREL(J,J)
116           FORMAT(' GREL(',I2,',',I2,') IN GROUP ',I2,' IS ',F8.3
   X,' .')
            CALL WRITEL(10)
            CALL HOLD
            GOTO 50
          ENDIF
117       CONTINUE
      ENDIF
      COUNT=NSEC(I)-1+COUNT
120   CONTINUE
    CALL WRITEL(25)
    WRITE(*,*) 'NEW METRIC TENSOR OF THE ENTIRE MOLECULE:'
    WRITE(*,*) '========================================='
    CALL WRITEL(2)
    DO 122 I=1,N-1
      WRITE(*,121) (GREL(I,J),J=1,N-1)
121     FORMAT(' ',20(F7.3,1X))
122   CONTINUE
```

```
      CALL WRITEL(3)
      WRITE(*,123) N-1
123   FORMAT(' IF ORTHONORMALIZATION IS COMPLETE (I.E. ABOVE MATRIX IS I
     X',I2,'.), USE OF THE GRAM')
      WRITE(*,*) 'SCHMIDT OR TOTALLY SYMMETRIC ORTHONORMALIZATION PROCED
     XURES WILL NOT CHANGE THE ORTHONORMAL VECTOR(S).'
      WRITE(*,*) '(THE ORDER OF PRIORITIES IN GRAM SCHMIDT WILL BE IRREL
     XEVANT.)'
      CALL WRITEL(2)
      WRITE(*,*) 'PRESS RETURN TO CONTINUE.'
      READ(*,124) ANS
124   FORMAT(A1)
      RETURN
      END
C
C
C
      SUBROUTINE TOPOL(TOPMAT,TCMREL,M,D,MAX20,WORK2)
C
C
      INTEGER*2 I,J,D,MAX20,K
      REAL*8 TCMREL(MAX20,1),M(MAX20,1),TOTMAS,NUM,WORK2(MAX20,1),
     X     TOPMAT(MAX20,1)
C
C
      NUM=0.0
      CALL MATSCA(WORK2,NUM,TOPMAT,D,MAX20)
      CALL ASSIGN(TOPMAT,WORK2,D,MAX20)
      CALL MATSCA(WORK2,NUM,TCMREL,D,MAX20)
      CALL ASSIGN(TCMREL,WORK2,D,MAX20)
      TOTMAS=M(1,1)
      DO 5 I=2,D
        TOTMAS=M(I,I) + TOTMAS
5     CONTINUE
7     CALL WRITEL(25)
      WRITE(*,*) 'THE TOPOLOGY MATRIX SPECIFIES THE VECTOR JOINING OF TH
     XE PARTICLES IN THE SOURCE'
      WRITE(*,*) 'COORDINATE SYSTEM. FOR N PARTICLES THERE WILL BE N-1 S
     XUCH VECTORS. A VECTOR'
      WRITE(*,*) 'JOINING TWO ATOMS I AND J IS DENOTED BY R(ij)=R(j)- R(
     Xi). IN RESPONSE TO EACH'
      WRITE(*,*) 'PROMPT, ENTER 1 IF THE VECTOR IS AS INDICATED AND -1 I
     XF IT HAS THE OPPOSITE'
      WRITE(*,*) 'DIRECTION. ENTER 0 IF THE ATOMS ARE UNCONNECTED.'


      WRITE(*,*)
      DO 25 I=1,D
        DO 20 J=1,D
10        IF (I .LT. J) THEN
            WRITE(*,15) I,J
15           FORMAT(' ',I1,'---->',I2)
```

```
          READ(*,*) TOPMAT(I,J)
          CALL WRITEL(1)
          IF ((TOPMAT(I,J) .NE. 1) .AND. (TOPMAT(I,J) .NE. 0)
     X         .AND. (TOPMAT(I,J) .NE. -1)) THEN
            CALL ERRORS
            GOTO 10
          ENDIF
        ENDIF
20    CONTINUE
25   CONTINUE
     K=0
     DO 50 I=1,(D-1)
       DO 40 J=1,D
         IF (TOPMAT(I,J) .NE. 0) THEN
           K=K+1
           IF (TOPMAT(I,J) .EQ. 1) THEN



             TCMREL(K,J)=1
             TCMREL(K,I)=-1
           ELSE
             TCMREL(K,J)=-1
             TCMREL(K,I)=1
           ENDIF
         ENDIF
40     CONTINUE
50   CONTINUE
     IF (K .LT. D-1) THEN
       CALL MATSCA(TOPMAT,NUM,WORK2,D,MAX20)
       CALL MATSCA(TCMREL,NUM,WORK2,D,MAX20)
       CALL WRITEL(3)
       WRITE(*,55) D-1
55      FORMAT(' ERROR !! YOU MUST SPECIFY AT LEAST ',I2,' BOND VECTORS
     X .')
       CALL WRITEL(3)
       CALL HOLD
       GOTO 7
     ENDIF
     DO 60 J=1,D
       TCMREL(D,J)=M(J,J)/TOTMAS
60   CONTINUE
     DO 80 I=1,D
       DO 70 J=1,D
         TOPMAT(I,J)=0
70     CONTINUE
80   CONTINUE
     RETURN
     END
```

```
C
C
C
      SUBROUTINE TRANS(WORK2,WORK4,GCMREL,GRAM,TCMREL,D,MAX20)
C
C
      INTEGER*2 I,J,D,MAX20
      REAL*8 WORK2(MAX20,1),WORK4(MAX20,1),GCMREL(MAX20,1),
     X     GRAM(MAX20,1),TCMREL(MAX20,1)
C
C
      CALL MATRAN(WORK2,TCMREL,D,MAX20)
      CALL MATMUL(WORK4,TCMREL,GRAM,D,MAX20)
      CALL MATMUL(GCMREL,WORK4,WORK2,D,MAX20)
      RETURN
      END
C
C
C
      SUBROUTINE TRUNC(D,GREL,CMNORM,GCMREL,MAX19,MAX20)
C
C
      INTEGER*2 D,MAX19,MAX20,I,J
      REAL*8 GREL(MAX19,1),CMNORM,GCMREL(MAX20,1)
C
C
      D=D-1
      CMNORM=1.0/SQRT(GCMREL(D+1,D+1))
      DO 20 I=1,D
        DO 10 J=1,D
          GREL(I,J)=GCMREL(I,J)
10      CONTINUE
20    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE GRCALC(GRAM,M,D,MAX20,WORK2,TOPMAT,TCMREL,WORK4,GCMREL,
     X                  GREL,CMNORM,MAX19)
C
C
C     THIS SUBROUTINE PRODUCES THE GRAM MATRIX FOR THE RELATIVE BASES.
C
      INTEGER*2 D,MAX19,MAX20
      REAL*8 GRAM(MAX20,1),M(MAX20,1),WORK2(MAX20,1),TOPMAT(MAX20,1),
     X     TCMREL(MAX20,1),WORK4(MAX20,1),GCMREL(MAX20,1),CMNORM,
     X     GREL(MAX19,1)
C
C
      CALL INVDET(GRAM,M,D,MAX20,DET,WORK2)
      CALL TOPOL(TOPMAT,TCMREL,M,D,MAX20,WORK2)
```

```
      CALL TRANS(WORK2,WORK4,GCMREL,GRAM,TCMREL,D,MAX20)
      CALL TRUNC(D,GREL,CMNORM,GCMREL,MAX19,MAX20)
      RETURN
      END
C
C
C
      SUBROUTINE TOPOL2(TOPMAT,TCMREL,M,D,MAX20,WORK2,L,METHOD,GNUM)
C
C
      INTEGER*2 I,J,D,MAX20,K,COUNT,L,GNUM
      REAL*8 TCMREL(MAX20,1),M(MAX20,1),TOTMAS,NUM,WORK2(MAX20,1),
     X     TOPMAT(MAX20,1)
      CHARACTER METHOD*1
C
C
      NUM=0.0
      CALL MATSCA(WORK2,NUM,TOPMAT,D,MAX20)
      CALL ASSIGN(TOPMAT,WORK2,D,MAX20)
      CALL MATSCA(WORK2,NUM,TCMREL,D,MAX20)
      CALL ASSIGN(TCMREL,WORK2,D,MAX20)
      TOTMAS=M(1,1)
      DO 4 I=2,D
        TOTMAS=M(I,I) + TOTMAS
4     CONTINUE
5     CALL WRITEL(25)
      IF ((METHOD .EQ. 'G') .OR. (METHOD .EQ. 'g')) THEN
        WRITE(*,*) 'CURRENT VECTOR(S) BELONG TO ENTIRE MOLECULE.'
      ENDIF
      IF ((METHOD .EQ. 'P') .OR. (METHOD .EQ. 'p')) THEN
        WRITE(*,10) L
10      FORMAT(' CURRENT VECTOR(S) BELONG TO GROUP ',I2,' .')
      ENDIF
      IF ((METHOD .EQ. 'C') .OR. (METHOD .EQ. 'c')) THEN
        WRITE(*,12) GNUM
12      FORMAT(' CURRENT VECTOR(S) ARE ',I2,' PARTICULAR ONES.')
      ENDIF
      CALL WRITEL(5)
      COUNT=0
      WRITE(*,15) D,D-1
15    FORMAT(' YOU HAVE NOW GENERATED ',I2,' ORTHONORMAL VECTORS. YOU NE
     XED TO SPECIFY ',I2,' VECTORS')
      WRITE(*,*) 'WHICH WILL BE ORTHOGONAL TO THE ''POLYSECTOR'' VECTOR.
     X (THE ''POLYSECTOR'' VECTOR'
      WRITE(*,*) 'IS OBTAINED AUTOMATICALLY VIA THE NEW TOPOLOGY MATRIX
     XWHICH IS TO BE SPE-'
      WRITE(*,20) D
20    FORMAT(' CIFIED.) THESE VECTORS WILL SIMPLY BE THE DIFFERENCE OF P
     XAIRS OF THE ',I2,' ORTHO-')
      WRITE(*,*) 'NORMAL VECTORS JUST GENERATED.'
      WRITE(*,*) 'IN RESPONSE TO EACH PROMPT, ENTER 1 IF THE DIFFERENCE
     XIS WHAT YOU WANT, AND -1'
      WRITE(*,*) 'IF THE OPPOSITE DIFFERENCE IS DESIRED. ENTER 0 IF YOU
```

```
    XDON''T WANT TO GENERATE A'
     WRITE(*,*) 'NEW VECTOR FROM THE INDICATED DIFFERENCE.'
     WRITE(*,*)
     DO 50 I=1,D
       DO 45 J=1,D
25       IF (I .LT. J) THEN
           WRITE(*,*) '->  ->'
           WRITE(*,30) J,I
30          FORMAT(' ',I2,' - ',I2)
           READ(*,*) TOPMAT(I,J)
           CALL WRITEL(1)
           IF ((TOPMAT(I,J) .NE. 1) .AND. (TOPMAT(I,J) .NE. 0)
     X          .AND. (TOPMAT(I,J) .NE. -1)) THEN
             CALL ERRORS
             GOTO 25
           ENDIF
           IF (TOPMAT(I,J) .NE. 0) THEN
             COUNT=COUNT+1
             WRITE(*,35) COUNT
35            FORMAT(' THE NEW VECTOR ',I2,' IS NOW SPECIFIED.')
             CALL WRITEL(2)
           ENDIF
         ENDIF
45     CONTINUE
50   CONTINUE
     IF (COUNT .LT. D-1) THEN
       CALL MATSCA(TOPMAT,NUM,WORK2,D,MAX20)
       CALL WRITEL(3)
       WRITE(*,55) D-1
55      FORMAT(' ERROR !! YOU MUST SPECIFY ',I2,' VECTOR(S).')
       CALL WRITEL(10)
       CALL HOLD
       GOTO 5
     ENDIF
     K=0
     DO 65 I=1,(D-1)
       DO 60 J=1,D
         IF (TOPMAT(I,J) .NE. 0) THEN
           K=K+1
           IF (TOPMAT(I,J) .EQ. 1) THEN
             TCMREL(K,J)=1
             TCMREL(K,I)=-1
           ELSE
             TCMREL(K,J)=-1
             TCMREL(K,I)=1
           ENDIF
         ENDIF
60     CONTINUE
65   CONTINUE
     DO 70 J=1,D
       TCMREL(D,J)=M(J,J)/TOTMAS
70   CONTINUE
     DO 80 I=1,D
```

```
      DO 75 J=1,D
        TOPMAT(I,J)=0
75     CONTINUE
80    CONTINUE
      RETURN
      END
C
C
C
$NOFLOATCALLS
$STORAGE:2
C
C
      SUBROUTINE SETUP(S,N,MAX20,MAX19,GREL,M,GRAM,TCMREL,WORK2,WORK4,
     X            GCMREL,I,METHOD,GNUM)
C
C
      INTEGER*2 D,I,N,CHOICE,MAX19,MAX20,GNUM,JOBN,IZ,IER,K
      REAL*8 M(MAX20,1),GRAM(MAX20,1),TCMREL(MAX20,1),WORK2(MAX20,1),
     X     NUM,WK(9),WORK4(MAX20,1),GCMREL(MAX20,1),GREL(MAX19,1),
     X     NMGREL(9,9),NMIZER(9,9),GRELIN(9,9),S2(9,9),S(MAX19,1),
     X     METRIC(45),WORK(9,9),GPERM(9,9),DETERM(10),TPERMU(9,9),
     X     WORK3(10,10),CMNORM,TOPMAT(10,10),DET,WORKB(9,9),
     X     DETMAT(9,9),SEQNCE(10,10),GPERMU(10,10)
      CHARACTER SNAME*20,METHOD*1
C
C
      NUM=0.0
5     CALL WRITEL(25)
      IF ((METHOD .EQ. 'P') .OR. (METHOD .EQ. 'p')) THEN
        WRITE(*,7) I
7       FORMAT(' ORTHONORMALIZATION OF GROUP ',I2,' VECTOR(S):')
      ENDIF
      IF ((METHOD .EQ. 'G') .OR. (METHOD .EQ. 'g')) THEN
        WRITE(*,*) 'ORTHONORMALIZATION OF MOLECULE''S VECTOR(S):'
      ENDIF
      IF ((METHOD .EQ. 'C') .OR. (METHOD .EQ. 'c')) THEN
        WRITE(*,8) GNUM
8       FORMAT(' ORTHONORMALIZATION OF ',I2,' PARTICULAR VECTOR(S):')
      ENDIF
      CALL WRITEL(3)
      WRITE(*,*) 'ORTHONORMALIZATION PROCEDURES AVAILABLE'
      WRITE(*,*) '======================================='
      WRITE(*,*)
      WRITE(*,*) '1.) TOTALLY SYMMETRIC'
      WRITE(*,*) '2.) GRAM SCHMIDT'
      WRITE(*,*) '3.) IRREDUCIBLE SYMMETRIC'
      WRITE(*,*)
      WRITE(*,*) 'CHOOSE A PROCEDURE.'
      CALL WRITEL(9)
      READ(*,*) CHOICE
```

```
      IF ((CHOICE .LT. 1) .OR. (CHOICE .GT. 3)) THEN
        CALL ERRORS
        GOTO 5
      ENDIF
      IF (CHOICE .EQ. 1) THEN
        CALL SYMM(NMGREL,NMIZER,GRELIN,S2,S,WORK,GREL,N,MAX19,WORKB)
        CALL MATSCA(S2,NUM,S,N,MAX19)
        CALL MATSCA(NMIZER,NUM,S,N,MAX19)
        CALL MATSCA(NMGREL,NUM,S,N,MAX19)
      ENDIF
      IF (CHOICE .EQ. 2) THEN
        CALL GRAMSC(GPERM,S2,DETMAT,SEQNCE,S,N,MAX19,MAX20,DETERM,
     X           GREL,WORK2,TPERMU,WORK3,WORK,GPERMU,WORKB,GRAM,
     X           I,METHOD,GNUM)
        CALL MATSCA(S2,NUM,S,N,MAX19)
        DO 20 J=1,D
          DETERM(J)=0.0
20      CONTINUE
        CALL MATSCA(SEQNCE,NUM,WORK2,N,MAX20)
      ENDIF
      IF (CHOICE .EQ. 3) THEN
        CALL IRRSYM(N,METRIC,GREL,WORK,S,MAX19,DETERM,MAX20,WK,S2,
     X           WORKB)
        CALL MATSCA(M,NUM,GRAM,N,MAX20)
        CALL MATSCA(S2,NUM,S,N,MAX19)
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE PSQRT(A,B,C,D,E,MAX,N,ERROR)
C
C
      INTEGER*2 I,J,K,MAX,N
      REAL*8 NUM,A(MAX,1),B(MAX,1),C(MAX,1),D(MAX,1),E(MAX,1)
      LOGICAL ERROR
C
C
      NUM=0.0
      CALL MATSCA(A,NUM,B,N,MAX)
      DO 15 I=1,470
        CALL ASSIGN(B,A,N,MAX)
        DO 10 J=1,N
          DO 5 K=1,N
            IF (ABS(B(J,K)) .GT. 1.0D20) THEN
              ERROR=.TRUE.
              GOTO 20
            ENDIF
5         CONTINUE
10      CONTINUE
        CALL MATMUL(C,B,A,N,MAX)
        CALL MATSUB(E,D,C,N,MAX)
```

```
      NUM=0.5
      CALL MATSCA(C,NUM,E,N,MAX)
      CALL ASSIGN(B,C,N,MAX)
      CALL MATADD(C,A,B,N,MAX)
      CALL ASSIGN(A,C,N,MAX)
15    CONTINUE
20    RETURN
      END
C
C
C
      SUBROUTINE NORM(D,NMIZER,GREL,NMGREL,MAX19)
C
C
      INTEGER*2 D,I,J,MAX19
      REAL*8 GREL(MAX19,1),NMIZER(MAX19,1),NMGREL(MAX19,1)
C
C
      DO 10 I=1,D
        NMIZER(I,I)=GREL(I,I)**(-1.0/2.0)
10    CONTINUE
      DO 30 I=1,D
        DO 20 J=1,D
          NMGREL(I,J)=GREL(I,J)/(GREL(I,I)*GREL(J,J))**(1.0/2.0)
20      CONTINUE
30    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE MINORS(K,DETMAT,GPERM,S2,MAX19,WORK,WORKB)
C
C
      INTEGER*2 I,J,A,K,MAX19
      REAL*8 DETMAT(MAX19,1),GPERM(MAX19,1),S2(MAX19,1),DET,
     X     WORK(MAX19,1),WORKB(MAX19,1)
C
C
2     DO 50 A=1,K
4       DO 20 I=1,K
6         DO 10 J=1,K
            DETMAT(I,J)=GPERM(I,J)
10        CONTINUE
20      CONTINUE
22      DO 30 J=1,K
          DETMAT(K,J)=0.0
30      CONTINUE
32      DO 40 I=1,K
          DETMAT(I,A)=0.0
40      CONTINUE
42      DETMAT(K,A)=1.0
44      CALL INVDET(WORK,DETMAT,K,MAX19,DET,WORKB)
```

```
46      S2(K,A)=DET
50    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE SYMM(NMGREL,NMIZER,GRELIN,S2,S,WORK,GREL,D,MAX19,WORKB)
C
C
      INTEGER*2 I,J,PROCED,D,MAX19
      REAL*8 NMGREL(MAX19,1),NMIZER(MAX19,1),GRELIN(MAX19,1),
     X       S2(MAX19,1),DET,S(MAX19,1),GREL(MAX19,1),WORK(MAX19,1),AC,
     X       NUM,WORKB(MAX19,1)
      LOGICAL ERROR
C
C
10    CALL WRITEL(25)
      WRITE(*,*) '1.) NORM INDEPENDENT SYMMETRIZATION (NORMALIZATION FOL
     XLOWED BY SYMMETRIZATION)'
      WRITE(*,*)
      WRITE(*,*) '2.) NORM DEPENDENT SYMMETRIZATION'
      CALL WRITEL(3)
      WRITE(*,*) 'CHOOSE A SYMMETRIZATION PROCEDURE.'
      CALL WRITEL(7)
      READ(*,*) PROCED
      IF ((PROCED .NE. 1) .AND. (PROCED .NE. 2)) THEN
        CALL ERRORS
        GOTO 10
      ENDIF
      ERROR=.FALSE.
      NUM=0.0
      CALL MATSCA(WORK,NUM,WORKB,D,MAX19)
      AC=1.0D-07
      DO 30 I=1,D
        NMIZER(I,I)=1
30    CONTINUE
      IF (PROCED .EQ. 1) THEN
        CALL NORM(D,NMIZER,GREL,NMGREL,MAX19)
        CALL INVDET(GRELIN,NMGREL,D,MAX19,DET,WORK)
      ELSE
        CALL INVDET(GRELIN,GREL,D,MAX19,DET,WORK)
      ENDIF
      IF (ABS(DET) .LT. AC) THEN
        WORK(1,1)=3.0
        GRELIN(1,1)=4.0
        GOTO 31
      ENDIF
      CALL PSQRT(S2,WORK,S,GRELIN,WORKB,MAX19,D,ERROR)
      IF (ERROR) THEN
        GOTO 31
      ENDIF
      CALL MATMUL(WORK,S2,S,D,MAX19)
```

```
31    DO 35 I=1,D
       DO 32 J=1,D
         IF (ABS(WORK(I,J)-GRELIN(I,J)) .GT. AC) THEN
           IF (PROCED .EQ. 2) THEN
             CALL PSQRT(S2,WORK,S,GREL,WORKB,MAX19,D,ERROR)
           ELSE
             CALL PSQRT(S2,WORK,S,NMGREL,WORKB,MAX19,D,ERROR)
           ENDIF
           CALL INVDET(S,S2,D,MAX19,DET,WORK)
           GOTO 65
         ENDIF
32      CONTINUE
35    CONTINUE
65    CALL MATMUL(WORK,S,NMIZER,D,MAX19)
      CALL ASSIGN(S,WORK,D,MAX19)
      RETURN
      END
C
C
C
      SUBROUTINE GRAMSC(GPERM,S2,DETMAT,SEQNCE,S,D,MAX19,MAX20,
     X                  DETERM,GREL,WORK2,TPERMU,WORK3,WORK,GPERMU,
     X                  WORKB,GRAM,L,METHOD,GNUM)
C
C
      INTEGER*2 A,I,J,K,D,MAX19,MAX20,VPRIOR(9),L,GNUM
      REAL*8 DET,GPERM(MAX19,1),S2(MAX19,1),DETMAT(MAX19,1),S(MAX19,1),
     X     DETERM(1),GREL(MAX19,1),WORK2(MAX20,1),TPERMU(MAX19,1),NUM,
     X     WORK3(MAX20,1),WORK(MAX19,1),SEQNCE(MAX20,1),
     X     GPERMU(MAX20,1),WORKB(MAX19,1),GRAM(MAX20,1)
      CHARACTER METHOD*1
C
C
      CALL WRITEL(25)
      IF ((METHOD .EQ. 'p') .OR. (METHOD .EQ. 'P')) THEN
        WRITE(*,15) L
15      FORMAT(' DEFINE THE SEQUENCE IN WHICH THE GROUP ',I2,' VECTOR(S
     X) WILL BE ORTHONORMALIZED.')
      ENDIF
      IF ((METHOD .EQ. 'G') .OR. (METHOD .EQ. 'g')) THEN
        WRITE(*,*) 'DEFINE THE SEQUENCE IN WHICH THE MOLECULE''S VECTOR
     X(S) WILL BE ORTHONORMALIZED.'
      ENDIF
      IF ((METHOD .EQ. 'C') .OR. (METHOD .EQ. 'c')) THEN
        WRITE(*,17) GNUM
17      FORMAT(' DEFINE THE SEQUENCE IN WHICH THE VECTOR(S) CORRESPONDI
     XNG TO THE ',I2,' PARTICULAR')
        WRITE(*,*) 'VECTOR(S) WILL BE ORTHONORMALIZED.'
      ENDIF
      CALL WRITEL(2)
```

```
      DO 35 I=D,1,-1
20      WRITE(*,25) I
25      FORMAT(' WHICH VECTOR HAS ORTHONORMALIZATION PRIORITY ',I2,' ?'
     X)
        READ(*,*) VPRIOR(I)
        WRITE(*,*)
        DO 30 J=(I+1),D
          IF (VPRIOR(I) .EQ. VPRIOR(J)) THEN
            CALL ERRORS
            GOTO 20
          ENDIF
30       CONTINUE
        IF ((VPRIOR(I) .LT. 1) .OR. (VPRIOR(I) .GT. D)) THEN
          CALL ERRORS
          GOTO 20
        ENDIF
35     CONTINUE
      CALL ASSIG3(GRAM,GREL,D,MAX20,MAX19)
      DETERM(1)=1.0
      DO 40 I=1,D
        J=VPRIOR(I)
        SEQNCE(I,J)=1
40     CONTINUE
      CALL INVDET(WORK2,SEQNCE,D,MAX20,DET,WORK3)
      CALL ASSIG3(TPERMU,WORK2,D,MAX19,MAX20)
      CALL MATMUL(WORK3,GRAM,WORK2,D,MAX20)
      CALL MATMUL(GPERMU,SEQNCE,WORK3,D,MAX20)
      CALL ASSIG3(GPERM,GPERMU,D,MAX19,MAX20)
41     DETERM(2) = GPERM(1,1)



      S2(1,1) = 1.0
42     DO 60 K=2,D
        DO 55 I=1,K
          DO 50 J=1,K
            GPERM(I,J)=GPERM(J,I)
50         CONTINUE
55       CONTINUE
        CALL INVDET(WORK,GPERM,K,MAX19,DET,WORKB)
        DETERM(K+1)=DET
        CALL MINORS(K,DETMAT,GPERM,S2,MAX19,WORK,WORKB)
60     CONTINUE
62     S2(1,1)=S2(1,1)/DETERM(2)**(1.0/2.0)
63     DO 70 K=2,D
64       DO 65 A=1,K
          S2(K,A)=S2(K,A)/(DETERM(K+1)*DETERM(K))**(1.0/2.0)
65       CONTINUE
70     CONTINUE
      CALL MATMUL(WORKB,TPERMU,S2,D,MAX19)
      CALL ASSIG3(S2,SEQNCE,D,MAX19,MAX20)
      CALL MATMUL(S,WORKB,S2,D,MAX19)
      RETURN
```

```
      END
C
C
C
      SUBROUTINE MATRAN(NEW,M,RANGE,SIZE)
C
C
      INTEGER*2 I,J,SIZE,RANGE
      REAL*8 NEW(SIZE,1),M(SIZE,1)
C
C
      DO 20 I=1,RANGE
        DO 10 J=1,RANGE
          NEW(I,J)=M(J,I)
10      CONTINUE
20    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE ASSIG3(NEW,M,RANGE,SIZE,SIZE2)
C
C
      INTEGER*2 SIZE,SIZE2,RANGE,I,J
      REAL*8 M(SIZE2,1),NEW(SIZE,1)
C
C
      DO 20 I=1,RANGE
        DO 10 J=1,RANGE
          NEW(I,J)=M(I,J)
10      CONTINUE
20    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE INVDET(AINV,A,RANGE,SIZE,DET,HOLD)
C
C
      INTEGER*2 K,RANGE,I,J,T,Q,SIZE,R,C,R1(40),C1(40),L,L1,L2,L3,L4,L5,
     X        L7
      REAL*8 DET,Y(40),A(SIZE,1),M,E,B,B2,HOLD(SIZE,1),P,AINV(SIZE,1)
      DO 5 I=1,RANGE
        DO 4 J=1,RANGE
          HOLD(I,J)=A(I,J)
4       CONTINUE
5     CONTINUE
      K=0
      DET=1
20    K=K+1
      IF (K .GT. RANGE) THEN
```

```
      GOTO 47
    ENDIF
    I=0
    J=0
    B=0
22   I=I+1
    IF (I .GT. RANGE) THEN
      GOTO 34
    ENDIF
    Q=0
24   Q=Q+1
    IF (Q .GT. (K-1)) THEN
      GOTO 27
    ENDIF
25   IF (I .EQ. R1(Q)) THEN
      GOTO 22
    ENDIF
    GOTO 24
27   J=J+1
    IF (J .GT. RANGE) THEN
      J=0
      GOTO 22
    ENDIF
    Q=0
29   Q=Q+1
    IF (Q .GT. (K-1)) THEN
      GOTO 32
    ENDIF
    IF (J .EQ. C1(Q)) THEN
      GOTO 27
    ENDIF
    GOTO 29
32   E= ABS(A(I,J))
    IF (E .GE. B) THEN
      R1(K)=I
      C1(K)=J
      B=E
    ENDIF
    GOTO 27
34   L7=C1(K)
    L=R1(K)
    B2=ABS(A(L,L7))
    P=A(L,L7)
    IF (B2 .LE. 0.000000000001) THEN
      DET=DET*P
      DO 36 I=1,RANGE
        DO 35 J=1,RANGE
          A(I,J)=HOLD(I,J)
35        CONTINUE
36      CONTINUE
      RETURN
    ENDIF
    DET=DET*P
```

```
    R=R1(K)
    C=C1(K)
    J=0
38   J=J+1
    IF (J .GT. RANGE) THEN
      GOTO 41
    ENDIF
    IF (J .EQ. C) THEN
      A(R,C)=1.0/P
      GOTO 38
    ENDIF
    A(R,J)=A(R,J)/P
    GOTO 38
41   I=0
    J=0
42   I=I+1
    IF (I .GT. RANGE) THEN
      GOTO 20
    ENDIF
    M=A(I,C)
    IF (I .EQ. R) THEN
      GOTO 42
    ENDIF
44   J=J+1
    IF (J .GT. RANGE) THEN
      J=0
      GOTO 42
    ENDIF
    IF (J .EQ. C) THEN
      A(I,C)=M/(-P*1.0)
      GOTO 44
    ENDIF
    A(I,J)=A(I,J)-A(R,J)*M
    GOTO 44
47   I=0
    J=0
48   J=J+1
    IF (J .GT. RANGE) THEN
      I=0
      J=0
      GOTO 53
    ENDIF
49   I=I+1
    IF (I .GT. RANGE) THEN
      I=0
      GOTO 51
    ENDIF
    L1=C1(I)
    L2=R1(I)
    Y(L1)=A(L2,J)
    GOTO 49
51   I=I+1
```

```
      IF (I .GT. RANGE) THEN
        I=0
        GOTO 48
      ENDIF
      A(I,J)=Y(I)
      GOTO 51
53    I=I+1
      IF (I .GT. RANGE) THEN
        GOTO 58
      ENDIF
54    J=J+1
      IF (J .GT. RANGE) THEN
        J=0
        GOTO 56
      ENDIF
      L3=R1(J)
      L4=C1(J)
      Y(L3)=A(I,L4)
      GOTO 54
56    J=J+1
      IF (J .GT. RANGE) THEN
        J=0
        GOTO 53
      ENDIF
      A(I,J)=Y(J)
      GOTO 56
58    I=0
59    I=I+1
      IF (I .LE. RANGE) THEN
        L5=R1(I)
        Y(L5)=C1(I)
        GOTO 59
      ENDIF
      I=0
61    I=I+1
      IF (I .GT. RANGE) THEN
        DO 63 J=1,RANGE
          DO 62 T=1,RANGE
            AINV(J,T)=A(J,T)
            A(J,T)=HOLD(J,T)
62        CONTINUE
63      CONTINUE
        RETURN
      ENDIF
      J=0
64    J=J+1
      IF (J .GT. (RANGE-1)) THEN
        GOTO 61
      ENDIF
65    IF (Y(J) .LE. Y(J+1)) THEN
        GOTO 64
      ENDIF
      E=Y(J)
```

```
      Y(J)=Y(J+1)
      Y(J+1)=E
      DET=(-DET)
      GOTO 64
      RETURN
      END
C
C
C
      SUBROUTINE HOLD
C
C
C     THIS SUBROUTINE CAUSES THE COMPUTER TO COUNT TO TOP AS A PAUSE.
C
C
      INTEGER*2 I,J,TOP
C
C
      TOP=150
C
C     PAUSE.
C
      DO 10 I=1,TOP
        DO 5 J=1,TOP
5       CONTINUE
10    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE WRITEL(NUMBER)
C
C
C     THIS SUBROUTINE PRINTS 'NUMBER' BLANK LINES.
C
C
      INTEGER*2 I,NUMBER
C
C
      DO 5 I=1,NUMBER
        WRITE(*,*)
5     CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE ERRORS
```

```
C
C
C    THIS SUBROUTINE PRINTS AN ERROR MESSAGE ON THE SCREEN AND THEN
C    PAUSES.
C
C
      CALL WRITEL(3)
      WRITE(*,*) '*** ERROR ***'
      WRITE(*,*)
      WRITE(*,*) 'INCORRECT INPUT !!!'
      CALL WRITEL(3)
      CALL HOLD
      RETURN
      END
C
C
C
      SUBROUTINE SAVEM2(HEAD1,HEAD2,HEAD3,ROWS,COLS,MATRIX,ROWNME,
     X                  COLNME,MAX)
C
C
C    THIS SUBROUTINE SAVES MATRIX TO DISK, BY FIRST ASKING FOR THE NAME
C    OF THE FILE TO STORE IT IN.
C
C
      INTEGER*2 TOP,ROWS,COLS,DISK,I,J,MAX
      REAL*8 MATRIX(MAX,1)
      CHARACTER HEAD1*20,HEAD2*20,HEAD3*20,COLNME(1)*5,ROWNME(1)*15,
     X       FILENA*20
C
C
      TOP=400
      IF (MAX .GT. TOP) THEN
        WRITE(*,*)
        WRITE(*,*) 'ERROR !!! ARRAY IS TOO LARGE.'
        CALL HOLD
        RETURN
      ENDIF
      DISK=5
      CALL WRITEL(2)
      WRITE(*,*) 'ENTER THE NAME OF THE FILE TO BE USED.'
      READ(*,5) FILENA
5     FORMAT(A20)
      OPEN(DISK,FILE=FILENA,STATUS='NEW')
      WRITE(DISK,10) HEAD1,HEAD2,HEAD3,ROWS,COLS
10    FORMAT(3(A20),2I4)
      DO 20,I=1,ROWS
        WRITE(DISK,15) (MATRIX(I,J),J=1,COLS)
15      FORMAT(400(F35.14))
20    CONTINUE
      DO 30,I=1,ROWS
        WRITE(DISK,25) ROWNME(I)
25      FORMAT(400(A15))
```

```
30   CONTINUE
     DO 40,J=1,COLS
       WRITE(DISK,35) COLNME(J)
35     FORMAT(400(A5))
40   CONTINUE
     CLOSE(DISK,STATUS='KEEP')
     RETURN
     END
C
C
C
     SUBROUTINE OUTPUT(SNAME,SMOLEC,MOLNAM,STOR1,MAX19,D,SECTIO,METHOD,
    X             GREL)
C
C
     INTEGER*2 CHOICE,VIDEO,PRINTE,FILENU,I,J,D,SECTIO
     REAL*8 SMOLEC(MAX19,1),STOR1(MAX19,1),GREL(MAX19,1)
     CHARACTER SNAME*20,MOLNAM*20,NAME*20,METHOD*1,ANS*1,ROWNME(9)*15,
    X       COLNME(9)*5
C
C
     VIDEO=0
     PRINTE=4
5    CALL WRITEL(25)
     WRITE(*,*) 'WHERE DO YOU WANT THE OUTPUT SENT ?'
     WRITE(*,*)
     WRITE(*,*) '1.) THE PRINTER'
     WRITE(*,*) '2.) THE SCREEN'
     CALL WRITEL(10)
     READ(*,*) CHOICE
     IF ((CHOICE .NE. 1) .AND. (CHOICE .NE. 2)) THEN
       CALL ERRORS
       GOTO 5
     ENDIF
     IF (CHOICE .EQ. 1) THEN
       OPEN(PRINTER,FILE='LPT1:')
       FILENU=PRINTE
     ELSE
       FILENU=VIDEO
     ENDIF
     DO 10 I=1,5
       WRITE(FILENU,*)
10   CONTINUE
     CALL WRITEL(25)
     WRITE(FILENU,*) '                    **********************
    X*****'
     WRITE(FILENU,15) MOLNAM
15   FORMAT('                       ** ',A20,' **')
     WRITE(FILENU,*) '                    **********************
    X*****'
     WRITE(FILENU,*)
```

```
      IF ((METHOD .EQ. 'G') .OR. (METHOD .EQ. 'g')) THEN
        WRITE(FILENU,*) '                    VECTORS TREATED G
     XLOBALLY'
      ELSE
        WRITE(FILENU,17) SECTIO
17     FORMAT('                VECTORS PARTITIONED INTO ',I2,
     X' GROUPS')
      ENDIF
      WRITE(FILENU,*)
      WRITE(FILENU,20) SNAME
20    FORMAT('         ORTHONORMALIZATION PROCEDURE(S) USED: ',A20)
      WRITE(FILENU,*)
      WRITE(FILENU,*) '                    **********************
     X*****'
      WRITE(FILENU,*) '                    ** FINAL METRIC TENSO
     XR **'
      WRITE(FILENU,*) '                    **********************
     X*****'
      WRITE(FILENU,*)
      DO 23 I=1,D
        WRITE(FILENU,22) (GREL(I,J),J=1,D)
22      FORMAT(' ',5X,20(F7.3,1X))
23    CONTINUE
      IF (FILENU .EQ. 0) THEN
        WRITE(FILENU,*)
        WRITE(FILENU,*)
        WRITE(*,*) 'PRESS RETURN TO SEE 0.'
        READ(*,24) ANS
24      FORMAT(A1)
        WRITE(FILENU,*)
        WRITE(FILENU,*)
      ELSE
        WRITE(FILENU,*)
        WRITE(FILENU,*)
      ENDIF
      WRITE(FILENU,*)
      WRITE(FILENU,*) '                              ********'
      WRITE(FILENU,*) '                              ** 0  **'
      WRITE(FILENU,*) '                              ********'
      WRITE(FILENU,*)
      DO 30 I=1,D
        WRITE(FILENU,25) (SMOLEC(I,J),J=1,D)
25      FORMAT(' ',5X,20(F7.3,1X))
30    CONTINUE
      IF (FILENU .EQ. 0) THEN
        WRITE(FILENU,*)
        WRITE(FILENU,*)
        WRITE(*,*) 'PRESS RETURN TO SEE 0^-1.'
        READ(*,35) ANS
35      FORMAT(A1)
        WRITE(FILENU,*)
        WRITE(FILENU,*)
```

```
      ELSE
        WRITE(FILENU,*)
        WRITE(FILENU,*)
      ENDIF
      WRITE(FILENU,*) '                              **********'
      WRITE(FILENU,*) '                              ** 0^-1 **'
      WRITE(FILENU,*) '                              **********'
      WRITE(FILENU,*)
      DO 45 I=1,D
        WRITE(FILENU,40) (STOR1(I,J),J=1,D)
 40     FORMAT(' ',5X,20(F7.3,1X))
 45   CONTINUE
      IF (FILENU .EQ. 0) THEN
        CALL WRITEL(2)
        WRITE(*,*) 'HIT RETURN TO SAVE 0 AND 0^-1 TO DISK.'
        READ(*,50) ANS
 50     FORMAT(A1)
      ELSE
        CLOSE(PRINTER)
      ENDIF
      CALL WRITEL(25)
      IF (FILENU .NE. 0) THEN
        CALL WRITEL(3)
        WRITE(*,*) 'PRINTING FINISHED ...'
        CALL WRITEL(10)
      ENDIF
      WRITE(*,*) 'SAVE 0 TO DISK.'
      I=D
      K=D
      WRITE(*,*)
      CALL SAVEM2(MOLNAM,NAME,SNAME,I,K,SMOLEC,ROWNME,COLNME,MAX19)
      CALL WRITEL(7)
      WRITE(*,*) 'SAVE 0^-1 TO DISK.'
      WRITE(*,*)
      NAME='    INVERSE'
      CALL SAVEM2(MOLNAM,NAME,SNAME,I,K,STOR1,ROWNME,COLNME,MAX19)
      RETURN
      END
C
C
C
      SUBROUTINE IRRSYM(N,METRIC,GREL,WORK,S,MAX19,DETERM,MAX20,WK,S2,
     X            WORKB)
C
C
      INTEGER*2 K,I,J,N,MAX19,MAX20
      REAL*8 METRIC(1),GREL(MAX19,1),NUM,WORK(MAX19,1),S(MAX19,1),AC,
     X     DETERM(1),S2(MAX19,1),WORKB(MAX19,1),WK(1)
C
C
      AC=1.0D-14
      NUM=0.0
      K=1
```

```
      DO 10 I=1,N
        DO 5 J=1,I
          WK(I)=0.0
          DETERM(I)=0.0
          IF (ABS(GREL(J,I)) .LT. AC) THEN
            GREL(J,I)=0.0
            METRIC(K)=0.0
          ELSE
            IF ((ABS(GREL(J,I)-1.0) .LT. AC) .AND. (I .EQ. J)) THEN
              METRIC(K)=1.0
              GREL(J,I)=1.0
            ELSE
              METRIC(K)=GREL(J,I)
            ENDIF
          ENDIF
          K=K+1
5       CONTINUE
10    CONTINUE
      CALL MATSCA(WORK,NUM,S,N,MAX19)
C
C     Put the call to a subroutine which calculates the eigenvectors
C     and eigenvalues of a real symmetric matrix here. If the subroutine
C     requires a matrix, use GREL, and if it requires a vector, use
C     METRIC (METRIC holds the upper triangular elements of GREL). Store
C     the eigenvectors in WORK, and the eigenvalues in DETERM.
C
      CALL XXXXXX(... )
C
C
      DO 20 I=1,N
        DO 15 J=1,N
          IF (I .NE. J) THEN
            WORKB(I,J)=0.0
          ELSE
            WORKB(I,I)=1.0/SQRT(DETERM(I))
          ENDIF
          S2(I,J)=WORK(J,I)
15      CONTINUE
20    CONTINUE
      CALL MATMUL(S,WORKB,S2,N,MAX19)
      RETURN
      END
```

Appendix B

Fortran–77 Source Code for *NPLOT*

```
$NOFLOATCALLS
$STORAGE:2
C
C
C
      PROGRAM NPLOT
C
C     THIS PROGRAM PLOTS CONTOUR LINES OF A PARTICULAR MOLECULE'S POTEN
C     TIAL ENERGY. ALL BUT TWO OF THE POTENTIAL'S VARIABLE COORDINATES
C     ARE FIXED. CURRENTLY, THE PROGRAM CAN BE APPLIED TO ONLY THREE AND
C     FOUR ATOM MOLECULES. SOME OF THE SUBROUTINES CALLED BY THE PROGRAM
C     CAN BE FOUND IN THE 'UTILITY PROGRAMS' APPENDIX.
C
C
      INTEGER*2 A,COLS,ROWS,CHOICE,I,MAX4,MAX5,MAX9,MAX42,N,NCOORD,
     X          VIDEO,STATUS
      INTEGER*4 SIZE,INTARY(2472)
      REAL*8 F(2,2),PARAM(5,9),XEQ(9),XMAX(9),H9(9),XFIR(9),V,
     X       XSEC(9),REQ(9),Q(9),MIN,QEQ(9),QMIN(9),QMAX(9),
     X       0(4,4),EMAX,ESTEP,HIGH,STEP,VEQ,XMIN(9),Y(4,4),
     X       TEMP1(4,4),TEMP2(4,4),X(9),TEMP3(4,4),TEMP4(4,4),
     X       EXTRA(4,4),MASSES(9),X2(9)
      CHARACTER ANS*1,DEVIC*7,MOLNAM*20,TRNAME*20,ONAME*20,COLNME(9)*5
     X          ROWNME(5)*15,LORB*1
      COMMON /GRACOM/ SIZE, INTARY
C
C
      DEVIC='DISPLAY'
      MAX4=4
```

```
      MAX5=5
      MAX9=9
      MAX42=2
      DO 2 I=1,MAX9
         Q(I)=1.0
2     CONTINUE
C
C     ASSIGN THE EQUILIBRIUM VALUES OF THE BOND DISTANCE COORDINATES TO
C     QEQ.
C
      CALL MOLPOT(QEQ,Q,V,MOLNAM,N,MASSES,LORB)
C
      IF ((N .GT. 5) .OR. (N .LE. 1)) THEN
         WRITE(*,3) N
3        FORMAT(' ERROR !!!   NPLOT CANNOT HANDLE A ',I2,' BODY PROBLEM.
     X.. PROGRAM WILL BE TERMINATED.')
         CALL HOLD
         GOTO 180
      ENDIF
      NCOORD=3*N-6
      VIDEO=0
C
C
      CALL OMAT(0,N,MAX4,ONAME,MASSES,MOLNAM,ROWNME,COLNME)
C
C
15    CALL WRITEL(25)
      WRITE(*,*) 'DO YOU WANT TO RECALL A PARAMETER MATRIX FROM DISK ? (
     XY/N)'
      CALL WRITEL(10)
      READ(*,20) ANS
20    FORMAT(A1)
      IF ((ANS .EQ. 'n') .OR. (ANS .EQ. 'N')) THEN
         DO 30 I=1,NCOORD
            Q(I)=QEQ(I)
30       CONTINUE
C
C        CALCULATE EQUILIBRIUM POTENTIAL VALUE.
C
         CALL MOLPOT(QEQ,Q,V,MOLNAM,N,MASSES,LORB)
C
C
         VEQ=V
         CALL QEQXEQ(QEQ,REQ,N,LORB,Y,MAX4,0,TEMP1,TEMP2,TEMP3,TEMP4,
     X              EXTRA,TRNAME,X)
         CALL ASSIGV(XEQ,X,NCOORD)
C
C        ASSIGN VALUES TO XMIN AND XMAX OF THE TARGET COORDINATES.
C
         DO 40 I=1,NCOORD
            IF (XEQ(I) .GT. 0.0) THEN
               XMIN(I)=0.01*XEQ(I)
               XMAX(I)=1.5*XEQ(I)
```

```
            ELSE
               IF (XEQ(I) .LT. 0.0) THEN
                  XMIN(I)=1.5*XEQ(I)
                  XMAX(I)=0.01*XEQ(I)
               ELSE
                  XMIN(I)=-0.7
                  XMAX(I)=0.7
               ENDIF
            ENDIF
40       CONTINUE
         CALL WRITEL(25)
         WRITE(*,*) 'GIVE THE UPPER ENERGY LIMIT FOR THE POTENTIAL ENERG
     XY PLOTS.'
         CALL WRITEL(5)
         READ(*,*) EMAX
         ESTEP=EMAX/10.0
         PARAM(1,1)=N
         PARAM(1,2)=NCOORD
         PARAM(1,3)=VEQ
         PARAM(1,4)=EMAX
         PARAM(1,5)=ESTEP
         DO 45 I=1,NCOORD
            PARAM(2,I)=QEQ(I)
            PARAM(3,I)=XEQ(I)
            PARAM(4,I)=XMIN(I)
            PARAM(5,I)=XMAX(I)
45       CONTINUE
         ROWS=MAX5
         IF (NCOORD .GE. 5) THEN
            COLS=NCOORD
         ELSE
            COLS=MAX5
         ENDIF
         COLNME(1)='N    '
         COLNME(2)='NCOOR'
         COLNME(3)='VEQ  '
         COLNME(4)='EMAX '
         COLNME(5)='ESTEP'
         ROWNME(2)='    QEQ'
         ROWNME(3)='    XEQ'
         ROWNME(4)='    XMIN'
         ROWNME(5)='    XMAX'
         CALL WRITEL(5)
         WRITE(*,*) 'SAVE THE PARAMETER MATRIX TO DISK.'
         CALL SAVEM2(MOLNAM,TRNAME,ONAME,ROWS,COLS,PARAM,ROWNME,COLNME,
     X              MAX5)
       ELSE
         IF ((ANS .EQ. 'y') .OR. (ANS .EQ. 'Y')) THEN
            CALL WRITEL(25)
            ROWS=MAX5
            IF (NCOORD .GE. 5) THEN
               COLS=NCOORD
```

```
            ELSE
               COLS=MAX5
            ENDIF
            WRITE(*,*) 'RECALL THE PARAMETER MATRIX FROM DISK.'
            CALL RECALL(PARAM,ROWNME,COLNME,MOLNAM,TRNAME,ONAME,MAX5,
     X                  ROWS,COLS)
            N=PARAM(1,1)
            NCOORD=PARAM(1,2)
            VEQ=PARAM(1,3)
            EMAX=PARAM(1,4)
            ESTEP=PARAM(1,5)
            DO 50 I=1,NCOORD
               QEQ(I)=PARAM(2,I)
               XEQ(I)=PARAM(3,I)
               XMIN(I)=PARAM(4,I)
               XMAX(I)=PARAM(5,I)
50          CONTINUE
         ELSE
            CALL ERRORS
            GOTO 15
         ENDIF
      ENDIF
60    CALL PLOT2(XEQ,N,NCOORD,XMAX,XMIN,XFIR,XSEC,H9,MAX42,F,MAX4,X,
     X           TEMP1,TEMP2,TEMP3,TEMP4,0,VEQ,EMAX,ESTEP,DEVIC,MOLNAM,
     X           TRNAME,ONAME,EXTRA,Y,QEQ,Q,MASSES,STATUS,X2,LORB,SIZE,
     X           INTARY)
      ROWS=MAX5
      IF (NCOORD .GE. 5) THEN
         COLS=NCOORD
      ELSE
         COLS=MAX5
      ENDIF
      CALL EDITMA(MOLNAM,TRNAME,ONAME,ROWS,COLS,PARAM,COLNME,ROWNME,
     X            VIDEO,MAX5)
      N=PARAM(1,1)
      NCOORD=PARAM(1,2)
      VEQ=PARAM(1,3)
      EMAX=PARAM(1,4)
      ESTEP=PARAM(1,5)
      DO 70 I=1,NCOORD
         QEQ(I)=PARAM(2,I)
         XEQ(I)=PARAM(3,I)
         XMIN(I)=PARAM(4,I)
         XMAX(I)=PARAM(5,I)
70    CONTINUE
170   CALL WRITEL(25)
      WRITE(*,*) 'DO YOU WANT ANOTHER PLOT ? (Y/N)'
      CALL WRITEL(10)
      READ(*,175) ANS
175   FORMAT(A1)
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
         DEVIC='DISPLAY'
         GOTO 60
```

```
      ELSE
         IF ((ANS .NE. 'N') .AND. (ANS .NE. 'n')) THEN
            CALL ERRORS
            GOTO 170
         ENDIF
      ENDIF
177   CALL WRITEL(25)
      WRITE(*,*) 'DO YOU WANT TO SAVE THE PARAMETER MATRIX TO DISK BEFOR
     XE NPLOT TERMINATION ?'
      WRITE(*,*) '(Y/N)'
      CALL WRITEL(10)
      READ(*,178) ANS
178   FORMAT(A1)
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
      - ROWS=MAX5
         IF (NCOORD .GE. 5) THEN
            COLS=NCOORD
         ELSE
            COLS=MAX5
         ENDIF
         CALL SAVEM2(MOLNAM,TRNAME,ONAME,ROWS,COLS,PARAM,ROWNME,COLNME,
     X              MAX5)
      ELSE
         IF ((ANS .NE. 'N') .AND. (ANS .NE. 'n')) THEN
            CALL ERRORS
            GOTO 177
         ENDIF
      ENDIF
180   STOP
      END
C
C
C
      SUBROUTINE OMAT(O,N,MAX4,ONAME,MASSES,MOLNAM,ROWNME,COLNME)
C
C
      INTEGER*2 ROWS,COLS,N,MAX4
      REAL*8 O(MAX4,1),MASSES(1)
      CHARACTER ANS*1,NAME*20,ROWNME(1)*15,COLNME(1)*5,ONAME*20,
     X          MOLNAM*20
C
C
C     RECALL AN O MATRIX FROM DISK.
C
      CALL WRITEL(25)
      WRITE(*,*) 'RECALL AN ''O'' MATRIX FROM DISK.'
      ROWS=N-1
      COLS=N-1
      NAME='      O MATRIX      '
      CALL RECALL(O,ROWNME,COLNME,MOLNAM,NAME,ONAME,MAX4,ROWS,COLS)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE COMPOT(X,N,S,F9,VEQ,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,
     X                  MOLNAM,EXTRA,TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
C
C
      INTEGER*2 N,MAX4
      REAL*8 X(1),V,Y(MAX4,1),QEQ(1),Q(1),F9,VEQ,S(MAX4,1),
     X       TEMP1(MAX4,1),TEMP2(MAX4,1),TEMP3(MAX4,1),TEMP4(MAX4,1),
     X       EXTRA(MAX4,1),MASSES(1)
      LOGICAL ERROR
      CHARACTER TRNAME*20,MOLNAM*20,LORB*1
C
C
C     CALCULATE THE TOTAL POTENTIAL.
C
      CALL FNV(X,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,TRNAM
     X         Y,QEQ,Q,MASSES,LORB,ERROR)
      F9=V-VEQ
      RETURN
      END
C
C
C
      SUBROUTINE FSOURC(Q,QSANG,N,LORB)
C
C
      INTEGER*2 N,I
      REAL*8 Q(1),QSANG(1),COSANG,PI
      CHARACTER LORB*1
C
C
C     CALCULATE THE BOND DISTANCE-BOND ANGLE COORDINATES FROM THE BOND
C     DISTANCE COORDINATES.
C
      PI=ACOS(-1.0)
      DO 5 I=1,(N-1)
         QSANG(I)=Q(I)
5     CONTINUE
      IF (N .EQ. 3) THEN
         COSANG=(Q(2)**2+Q(1)**2-Q(3)**2)/(2.0*Q(2)*Q(1))
         CALL ACHECK(COSANG)
         QSANG(3)=ACOS(COSANG)
      ENDIF
      IF ((N.EQ. 4) .AND. (LORB .EQ. 'B')) THEN
         COSANG=(Q(1)**2+Q(2)**2-Q(4)**2)/(2.0*Q(2)*Q(1))
         CALL ACHECK(COSANG)
         QSANG(4)=ACOS(COSANG)
         COSANG=(Q(1)**2+Q(3)**2-Q(5)**2)/(2.0*Q(1)*Q(3))
         CALL ACHECK(COSANG)
         QSANG(5)=ACOS(COSANG)
         COSANG=(Q(2)**2+Q(3)**2-Q(6)**2)/(2.0*Q(2)*Q(3))
```

```
         CALL ACHECK(COSANG)
         QSANG(6)=ACOS(COSANG)
      ENDIF
      IF ((N .EQ. 4) .AND. (LORB .EQ. 'L')) THEN
         COSANG=(Q(1)**2+Q(2)**2-Q(4)**2)/(2.0*Q(2)*Q(1))
         CALL ACHECK(COSANG)
         QSANG(4)=ACOS(COSANG)
         COSANG=(-Q(5)**2+Q(1)**2+Q(3)**2)/(2.0*Q(1)*Q(3))
         CALL ACHECK(COSANG)
         QSANG(5)=PI-ACOS(COSANG)
         COSANG=(Q(4)**2+Q(5)**2-Q(6)**2-Q(1)**2)/(2.0*Q(2)*Q(3))
         CALL ACHECK(COSANG)
         QSANG(6)=ACOS(COSANG)
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE TSOURC(QSANG,Q,N,LORB,ERROR)
C
C
C     CALCULATE THE BOND DISTANCE COORDINATES FROM THE BOND DISTANCE
C     -BOND ANGLE COORDINATES.
C


      INTEGER*2 N
      REAL*8 QSANG(1),Q(1),PI,AC
      LOGICAL ERROR
      CHARACTER LORB*1
C
C
      AC=1.0D-08
      PI=ACOS(-1.0)
      DO 5 I=1,(N-1)
         Q(I)=QSANG(I)
5     CONTINUE
      IF (N .EQ. 3) THEN
         Q(3)=(QSANG(1)**2.0+QSANG(2)**2.0-2.0*QSANG(1)*QSANG(2)*
     X        COS(QSANG(3)))**(1.0/2.0)
      ENDIF
      IF (N .EQ. 4) THEN
         IF (LORB .EQ. 'L') THEN
            IF ((QSANG(4) .EQ. PI) .AND. (QSANG(6) .NE.
     X          (PI - QSANG(5)))) THEN
               ERROR=.TRUE.
               WRITE(*,*) 'ERROR !! INCORRECT CONFIGURATION OF MOLECULE.
     X'
               GOTO 10
            ENDIF
```

```
            IF ((QSANG(5) .EQ. 0.0) .AND. (QSANG(4) .NE. QSANG(6))) THEN
               ERROR=.TRUE.
               WRITE(*,*) 'ERROR !! INCORRECT CONFIGURATION OF MOLECULE.
     X'
               GOTO 10
            ENDIF
         ENDIF
C
C        CHECK THAT ANGLES OBEY 'GEOMETRY RULES.'
C
         IF (QSANG(4)+QSANG(6)+QSANG(5)-2.0*PI .GT. -AC) THEN
            ERROR=.TRUE.
            WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
            GOTO 10
         ENDIF
         IF (QSANG(4)+QSANG(6)-QSANG(5) .LT. -AC) THEN
            ERROR=.TRUE.
            WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
            GOTO 10
         ENDIF
         IF (QSANG(4)+QSANG(5)-QSANG(6) .LT. -AC) THEN
            ERROR=.TRUE.
            WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
            GOTO 10
         ENDIF
         IF (QSANG(6)+QSANG(5)-QSANG(4) .LT. -AC) THEN
            ERROR=.TRUE.
            WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
            GOTO 10
         ENDIF
         IF ((N .EQ. 4) .AND. (LORB .EQ. 'B')) THEN
            Q(4)=(QSANG(1)**2.0+QSANG(2)**2.0-2.0*QSANG(1)*QSANG(2)*
     X           COS(QSANG(4)))**(1.0/2.0)
            Q(5)=(QSANG(1)**2.0+QSANG(3)**2.0-2.0*QSANG(1)*QSANG(3)*
     X           COS(QSANG(5)))**(1.0/2.0)
            Q(6)=(QSANG(2)**2.0+QSANG(3)**2.0-2.0*QSANG(2)*QSANG(3)*
     X           COS(QSANG(6)))**(1.0/2.0)
         ENDIF
         IF ((N .EQ. 4) .AND. (LORB .EQ. 'L')) THEN
            Q(4)=(QSANG(1)**2.0+QSANG(2)**2.0-2.0*QSANG(1)*QSANG(2)*
     X           COS(QSANG(4)))**(1.0/2.0)
            Q(5)=(Q(1)**2.0+Q(3)**2.0-2.0*QSANG(1)*QSANG(3)*
     X           COS(PI-QSANG(5)))**(1.0/2.0)
           Q(6)=(Q(4)**2.0+Q(5)**2.0-QSANG(1)**2.0-
     X           2.0*QSANG(2)*QSANG(3)*COS(QSANG(6)))**(1.0/2.0)
         ENDIF
       ENDIF
10     RETURN
       END
```

```
C
C
C
      SUBROUTINE PLOT2(XEQ,N,NCOORD,XMAX,XMIN,XFIR,XSEC,H9,MAX42,F,MAX
     X                 X,TEMP1,TEMP2,TEMP3,TEMP4,S,VEQ,EMAX,ESTEP,DEVIC,
     X                 MOLNAM,TRNAME,SNAME,EXTRA,Y,QEQ,Q,MASSES,STATUS,
     X                 X2,LORB,SIZE,INTARY)
C
C
      INTEGER*2 MAX43,MAX42,POINTS,N,NCOORD,CHOICE,I,J,FIR,SEC,MAX4,
     X          STATUS,COUNT,COUNT2,ROWS,COLS
      INTEGER*4 SIZE,INTARY(1)
      REAL*8 XEQ(1),XFIR(1),XSEC(1),H9(1),F9,Y(MAX4,1),XMIN(1),XMAX(1)
     X       F(MAX42,1),X(1),TEMP2(MAX4,1),TEMP3(MAX4,1),TEMP4(MAX4,1),
     X       STEP,S(MAX4,1),VEQ,EMAX,MIN,MAX,ESTEP,TEMP1(MAX4,1),
     X       EXTRA(MAX4,1),QEQ(1),Q(1),MASSES(1),X2(1)
      LOGICAL ERROR
      CHARACTER PONAME*20,ANS*1,DEVIC*7,MOLNAM*20,TRNAME*20,SNAME*20,
     X          LORB*1
C
C
      ERROR=.FALSE.
      MAX43=MAX42+1
5     CALL WRITEL(25)
      WRITE(*,10) MAX43
10    FORMAT(' GIVE THE NUMBER OF GRID POINTS (1 < POINTS < ',I2,').')
      CALL WRITEL(10)
      READ(*,*) POINTS
      IF ((POINTS .GT. MAX42) .OR. (POINTS .LT. 2)) THEN
         CALL ERRORS
         GOTO 5
      ENDIF
      DO 15 I=1,NCOORD
         H9(I)=(XMAX(I)-XMIN(I))/(POINTS-1)
15    CONTINUE
20    CALL WRITEL(25)
      WRITE(*,*) 'WHICH TWO COORDINATES (IN ORDER) ARE VARIABLE ?'
      CALL WRITEL(10)
      READ(*,*) FIR,SEC
      IF ((FIR .LT. 1) .OR. (FIR .GT. NCOORD)) THEN
         CALL ERRORS
         GOTO 20
      ENDIF
      IF ((SEC .LT. 1) .OR. (SEC .GT. NCOORD)) THEN
         CALL ERRORS
         GOTO 20
      ENDIF
25    CALL WRITEL(25)
      WRITE(*,*) 'CHOOSE A POTENTIAL TYPE FOR PLOTTING.'
      CALL WRITEL(2)
      WRITE(*,*) '(1.) COMPLETE'
      WRITE(*,*) '(2.) SEPARABLE'
      WRITE(*,*) '(3.) NONSEPARABLE'
```

```
      CALL WRITEL(10)
      READ(*,*) CHOICE
      IF ((CHOICE .GT. 3) .OR. (CHOICE .LT. 1)) THEN
        CALL ERRORS
        GOTO 25
      ENDIF
      CALL WRITEL(25)
      WRITE(*,27) TRNAME
27    FORMAT(' CALCULATING POTENTIAL IN ',A20,' COORDINATES ...')
      CALL WRITEL(10)
      IF (CHOICE .EQ. 1) THEN
        PONAME='TOTAL               '
      ELSE
        IF (CHOICE .EQ. 2) THEN
          PONAME='SEPARABLE          '
        ELSE
          PONAME='NON-SEPARABLE      '
        ENDIF
      ENDIF
C
C     MOVE THROUGH RANGE OF 'Y' AXIS COORDINATE.
C
      DO 35 I=1,POINTS
C
C       MOVE THROUGH RANGE OF 'X' AXIS COORDINATE.
C
        DO 30 J=1,POINTS
          COUNT=POINTS - I + 1
          COUNT2=POINTS - J + 1
          WRITE(*,28) COUNT,COUNT2
28        FORMAT('+',I2,'  ',I2)
          CALL ASSIGV(X,XEQ,NCOORD)
          CALL ASSIGV(XSEC,XEQ,NCOORD)
          CALL ASSIGV(XFIR,XEQ,NCOORD)
          XFIR(FIR)=(J-1)*H9(FIR) + XMIN(FIR)
          XSEC(SEC)=(I-1)*H9(SEC) + XMIN(SEC)
          X(FIR)=XFIR(FIR)
          X(SEC)=XSEC(SEC)
          IF (CHOICE .EQ. 1) THEN
            CALL COMPOT(X,N,S,F9,VEQ,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,
     X                  MOLNAM,EXTRA,TRNAME,Y,QEQ,Q,MASSES,LORB,
     X                  ERROR)
          ELSE
            IF (CHOICE .EQ. 2) THEN
              CALL SEPTAR(X,Q,QEQ,N,S,F9,VEQ,MAX4,TEMP1,TEMP2,TEMP3,
     X                    TEMP4,FIR,Y,XFIR,SEC,XSEC,XEQ,MOLNAM,
     X                    EXTRA,TRNAME,MASSES,LORB,ERROR)
            ELSE
              CALL ASSIGV(X2,X,NCOORD)
              CALL DIFF(X,Q,QEQ,N,S,F9,VEQ,MAX4,TEMP1,TEMP2,TEMP3,
     X                  TEMP4,FIR,Y,XFIR,SEC,XSEC,XEQ,MOLNAM,EXTRA,
     X                  TRNAME,MASSES,X2,LORB,ERROR)
            ENDIF
```

```
            ENDIF
            IF (ERROR) THEN
               GOTO 110
            ENDIF
            F(I,J)=F9
30       CONTINUE
35     CONTINUE
36     CALL WRITEL(25)
      WRITE(*,*) 'DO YOU WANT THE POTENTIAL MATRIX PRINTED ON THE SCREEN
     X ? (Y/N)'
      CALL WRITEL(10)
      READ(*,37) ANS
37     FORMAT(A1)
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
         CALL WRITEL(25)
         WRITE(*,*) 'POTENTIAL MATRIX:'
         CALL WRITEL(3)
         DO 45 I=1,POINTS
            WRITE(*,40) (F(I,J), J=1,POINTS)
40           FORMAT(' ',100(F9.1,1X))
45       CONTINUE
         CALL WRITEL(2)
         WRITE(*,*) 'PRESS RETURN TO CONTINUE.'
         READ(*,47) ANS
47       FORMAT(A1)
      ELSE
         IF ((ANS .NE. 'N') .AND. (ANS .NE. 'n')) THEN
            CALL ERRORS
            GOTO 36
         ENDIF
      ENDIF
48     CALL WRITEL(25)
      WRITE(*,*) 'DO YOU WANT TO SAVE THE POTENTIAL MATRIX TO DISK ? (Y/
     XN)'
      CALL WRITEL(10)
      READ(*,49) ANS
49     FORMAT(A1)
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
         ROWS=POINTS
         COLS=POINTS
         CALL WRITEL(25)
         WRITE(*,*) 'SAVE THE POTENTIAL MATRIX TO DISK.'
         CALL ASYSTA(ROWS,COLS,F,MAX42)
      ELSE
         IF ((ANS .NE. 'N') .AND. (ANS .NE. 'n')) THEN
            CALL ERRORS
            GOTO 48
         ENDIF
      ENDIF
      IF (CHOICE .EQ. 3) THEN
         EMAX=7000.0
         ESTEP=1000.0
         MIN=-7000.0
```

```
      ELSE
         MIN=0.0
C
C        FIND THE MINIMUM VALUE OF THE POTENTIAL VALUES CALCULATED.
C
         DO 55 I=1,POINTS
            DO 50 J=1,POINTS
               IF ((F(I,J)) .LT. MIN) THEN
                  MIN=F(I,J)
               ENDIF
50          CONTINUE
55       CONTINUE
      ENDIF
60    CALL DRAW(FIR,SEC,XMAX,XMIN,ESTEP,MIN,EMAX,MAX42,POINTS,H9,F,
     X          NCOORD,DEVIC,MOLNAM,TRNAME,PONAME,SNAME,XEQ,STATUS,SIZE,
     X          INTARY)
      CALL WRITEL(25)
65    WRITE(*,*) 'DO YOU WANT A HARD COPY ? (Y/N)'
      CALL WRITEL(10)
      READ(*,100) ANS
100   FORMAT(A1)
      IF ((ANS .EQ. 'Y') .OR. (ANS .EQ. 'y')) THEN
102      CALL WRITEL(25)



         WRITE(*,*) 'CHOOSE DEVICE.'
         WRITE(*,*)
         WRITE(*,*) '1.) PRINTER'
         WRITE(*,*) '2.) PLOTTER'
         CALL WRITEL(10)
         READ(*,105) ANS
105      FORMAT(A1)
         IF (ANS .EQ. '1') THEN
            DEVIC='PRINTER'
         ELSE
            IF (ANS .EQ. '2') THEN
               DEVIC='PLOTTER'
            ELSE
               CALL ERRORS
               GOTO 102
            ENDIF
         ENDIF
         GOTO 60
      ELSE
         IF ((ANS .NE. 'N') .AND. (ANS .NE. 'n')) THEN
            CALL ERRORS
            GOTO 65
         ENDIF
      ENDIF
110   RETURN
      END
```

```
C
C
C
      SUBROUTINE ASYSTA(ROWS,COLS,MATRIX,MAX)
C
C
      INTEGER*2 TOP,ROWS,COLS,DISK,I,J,MAX
      REAL*8 MATRIX(MAX,1)
      CHARACTER FILENA*20
C
C     THIS SUBROUTINE SAVES MATRIX TO DISK, BY FIRST ASKING FOR THE NAME
C     OF THE FILE TO STORE IT IN.
C
      TOP=400
      IF (MAX .GT. TOP) THEN
         WRITE(*,*)
         WRITE(*,*) 'ERROR !!! ARRAY IS TOO LARGE.'
         CALL HOLD
         RETURN
      ENDIF
      DISK=5
      CALL WRITEL(5)
      WRITE(*,*) 'ENTER THE NAME OF THE FILE TO BE USED.'
      READ(*,5) FILENA
5     FORMAT(A20)
      OPEN(DISK,FILE=FILENA,STATUS='NEW')
      DO 20,I=1,ROWS
         WRITE(DISK,15) (MATRIX(I,J),J=1,COLS)
15       FORMAT(400(F8.1,','))
20    CONTINUE
      CLOSE(DISK,STATUS='KEEP')
      RETURN
      END
C
C
C
      SUBROUTINE DRAW(FIR,SEC,XMAX,XMIN,ESTEP,MIN,EMAX,MAX42,POINTS,H9
     X               F7,NCOORD,DEVIC,MOLNAM,TRNAME,PONAME,SNAME,XEQ,
     X               STATUS,SIZE,INTARY)
C
C
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER*4 ISTEP,IMIN,IMAX,F,SIZE,INTARY(1)
      REAL*8 X1(7),Y1(7),X,H9(1),XMAX(1),XMIN(1),Y,F0,F1,F2,F3,
     X       F7(MAX42,1),XEQ(1),ESTEP,MIN,EMAX
      REAL DD,EE,FF,GG,HH,II,P(4),NUM,XRAY(11),YRAY(11)
      CHARACTER DEVIC*7,MOLNAM*20,TRNAME*20,PONAME*20,SNAME*20,ANS*1,
     X         TITLE(4)*5,STRIN1*21,STRIN2*21,VAL1*1,VAL2*1,
     X         VALUES(9)*7,STRIN3*29,STRIN4*30,GRIDP*14
      LOGICAL FLAG,FLAG2,FLAG3
```

```
C
C
C     DRAW AXIS, LABELS, AND TITLES FOR PARTICULAR POTENTIAL.
C
      ISTEP=IDNINT(ESTEP)
      IMIN=IDNINT(MIN)
      IMAX=IDNINT(EMAX)
      MODE=3
      CALL QSMODE(MODE)
      STATUS=POPNPS()
      STATUS = PPSOTS(7,DEVIC)
2     DATA XRAY /1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0,
     X          10.0, 11.0/
      DATA YRAY /1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0,
     X          10.0, 11.0/
      DD=10.0
      EE=10.0
      STATUS = PSSURF(DD,EE)
      STATUS = PDSXY(1,11,XRAY,YRAY)
      STATUS = PDSVIS(1,0)
      STATUS = PAXLBS(1,24,'/ / / / / / / / / / / //')
      STATUS = PAXLBS(2,24,'/ / / / / / / / / / / //')
      STATUS = PSNCLR(3)
      STATUS = PSNHGT(2)
      WRITE(STRIN1,5) IMIN
5     FORMAT('V(MIN.)=',I6,' CM.^-1')
      WRITE(STRIN2,7) IMAX
7     FORMAT('V(MAX.)=',I6,' CM.^-1')
      WRITE(STRIN4,9) ISTEP
9     FORMAT('CONTOUR INTERVAL=',I6,' CM.^-1')
      WRITE(GRIDP,10) POINTS,POINTS
10    FORMAT(I2,' X ',I2,' POINTS')
      STATUS = PNOTES(1,0.0,97.0,9,'MOLECULE:')
      STATUS = PNOTES(1,0.0,94.0,20,MOLNAM)
      STATUS = PNOTES(1,0.0,77.0,15,'POTENTIAL TYPE:')
      STATUS = PNOTES(1,0.0,74.0,20,PONAME)
      STRIN3 = 'ORTHONORMALIZATION PROCEDURE:'
      STATUS = PNOTES(1,0.0,19.3,29,STRIN3)
      STATUS = PNOTES(1,58.0,19.3,20,SNAME)
      STATUS = PNOTES(1,0.0,15.6,15,'TRANSFORMATION:')
      STATUS = PNOTES(1,30.0,15.6,20,TRNAME)
      STATUS = PNOTES(1,0.0,10.0,21,STRIN1)
      STATUS = PNOTES(1,0.0,6.0,21,STRIN2)
      STATUS = PNOTES(1,0.0,2.0,30,STRIN4)
      STATUS = PNOTES(1,65.0,2.0,14,GRIDP)
      NUM=60.0
      DO 30 I=1,NCOORD
        IF ((I .NE. FIR) .AND. (I .NE. SEC)) THEN
          WRITE(VAL1,12) I
12        FORMAT(I1)
          WRITE(VALUES(I),15) XEQ(I)
15        FORMAT(F7.3)
          STATUS=PNOTES(1,0.0,NUM,1,'X')
```

```
            STATUS=PNOTES(1,1.9,NUM,1,VAL1)
            STATUS=PNOTES(1,3.8,NUM,1,'=')
            STATUS=PNOTES(1,6.6,NUM,7,VALUES(I))
            NUM=NUM-6.0
          ENDIF
30     CONTINUE
       FF=29.0
       GG=23.0
       HH=100.0
       II=100.0
       STATUS = PSVIEW(FF,GG,HH,II)
       STATUS = PTAHGT(1,4)
       STATUS = PTAHGT(2,4)
       WRITE(VAL1,40) FIR
40     FORMAT(I1)
       WRITE(VAL2,50) SEC
50     FORMAT(I1)
       STATUS = PTAXS(1,1,' ')
       STATUS = PXCFRM(1,1)
       STATUS = PPLTIT()
       STATUS = PSNCLR(1)
       STATUS = PSNHGT(3)
       STATUS = PNOTES(1,47.9,5.1,1,'X')
       STATUS = PNOTES(1,1.0,80.0,1,'X')
       STATUS = PNOTES(1,50.9,5.1,1,VAL1)
       STATUS = PNOTES(1,4.0,80.0,1,VAL2)
       WRITE(TITLE(1),60) XMIN(FIR)
60     FORMAT(F5.2)
       WRITE(TITLE(2),70) XMIN(SEC)
70     FORMAT(F5.2)
       WRITE(TITLE(3),80) XMAX(FIR)
80     FORMAT(F5.2)
       WRITE(TITLE(4),90) XMAX(SEC)
90     FORMAT(F5.2)
       STATUS = PNOTES(1,6.0,5.1,5,TITLE(1))
       STATUS = PNOTES(1,0.0,9.5,5,TITLE(2))
       STATUS = PNOTES(1,84.0,5.1,5,TITLE(3))
95     STATUS = PNOTES(1,0.0,95.0,5,TITLE(4))
C
C      DRAW CONTOUR PLOTS OF THE POTENTIAL STORED IN F7.
C
96     Z=0
       DO 930 F = IMIN,IMAX,ISTEP
          STATUS = PSLCLR(1)
          IF (F .LT. 0) THEN
             STATUS = PSLCLR(2)
          ENDIF
          DO 927 I=1,(POINTS-1)
             X= (I-1)*H9(FIR)+XMIN(FIR)
             DO 920 J = 1,(POINTS-1)
                Y=(J-1)*H9(SEC)+XMIN(SEC)
                FLAG=.TRUE.
                F0=F7(J,I)-F
```

```
            F1=F7(J,I+1)-F
            F2=F7(J+1,I)-F
            F3=F7(J+1,I+1)-F
            C=1
            C1=0
            IF ((F0*F1) .LE. 0.0) THEN
               X1(C) = (((X*F1-(X+H9(FIR))*F0)/(F1-F0))-XMIN(FIR))*
     X               (100/(XMAX(FIR)-XMIN(FIR)))
               Y1(C) = (Y-XMIN(SEC))*(100/(XMAX(SEC)-XMIN(SEC)))
               C=C+1
            ENDIF
            IF ((F2*F0) .LE. 0.0) THEN
               X1(C)=(X-XMIN(FIR))*(100/(XMAX(FIR)-XMIN(FIR)))
               Y1(C)=(((Y*F2-(Y+H9(SEC))*F0)/(F2-F0))-XMIN(SEC))*
     X               (100/(XMAX(SEC)-XMIN(SEC)))
               C=C+1
            ENDIF
            IF ((F3*F1) .LE. 0.0) THEN
               X1(C)=((X+H9(FIR))-XMIN(FIR))*(100/(XMAX(FIR)-
     X               XMIN(FIR)))
               Y1(C)=(((Y*F3-(Y+H9(SEC))*F1)/(F3-F1))-XMIN(SEC))*
     X               (100/(XMAX(SEC)-XMIN(SEC)))
               C=C+1
            ENDIF
            IF ((F3*F2) .LE. 0.0) THEN
               C1=1
               X1(C)=(((X*F3-(X+H9(FIR))*F2)/(F3-F2))-XMIN(FIR))*
     X               (100/(XMAX(FIR)-XMIN(FIR)))
               Y1(C)=((Y+H9(SEC))-XMIN(SEC))*(100.0/(XMAX(SEC)-
     X               XMIN(SEC)))
               C=C+1
            ENDIF
            IF (C .NE. 1) THEN
               FLAG2=.TRUE.
               IF (C .NE. 5) THEN
                  FLAG2=.FALSE.
                  IF (Z .NE. 1) THEN
                     P(1)=REAL(X1(1))
                     P(2)=REAL(Y1(1))
                     P(3)=REAL(X1(2))
                     P(4)=REAL(Y1(2))
                     STATUS = PPLINE(2,2,P)
                     Z=1
                     FLAG=.FALSE.
                  ENDIF
                  IF (FLAG) THEN
                     P(1)=P(3)
                     P(2)=P(4)
                     P(3)=REAL(X1(1))
                     P(4)=REAL(Y1(1))
                     STATUS = PPLINE(2,2,P)
                     P(1)=P(3)
                     P(2)=P(4)
```

```
            P(3)=REAL(X1(2))
            P(4)=REAL(Y1(2))
            STATUS = PPLINE(2,2,P)
            Z=1
         ENDIF
         IF (C1 .NE. 1) THEN
            Z=0
         ENDIF
      ENDIF
      IF (FLAG2) THEN
         FLAG3=.TRUE.
         IF (Z .NE. 1) THEN
            FLAG3=.FALSE.
            P(1)=REAL(X1(1))
            P(2) =REAL( Y1(1))
            P(3) =REAL( X1(2))
            P(4) =REAL( Y1(2))
            STATUS = PPLINE(2,2,P)
            P(3) =REAL( X1(3))
            P(4) =REAL( Y1(3))
            STATUS = PPLINE(2,2,P)
            P(3) =REAL(X1(4))
            P(4) =REAL(Y1(4))
            STATUS = PPLINE(2,2,P)
            P(1)=REAL(X1(2))
            P(2) =REAL(Y1(2))
            P(3) =REAL( X1(3))
            P(4) =REAL( Y1(3))
            STATUS = PPLINE(2,2,P)
            P(3) =REAL( X1(4))
            P(4) =REAL( Y1(4))
            STATUS = PPLINE(2,2,P)
            P(1)=REAL( X1(3))
            P(2) =REAL( Y1(3))
            P(3) =REAL( X1(4))
            P(4) =REAL( Y1(4))
            STATUS = PPLINE(2,2,P)
         ENDIF
         IF (FLAG3) THEN
            P(1)= P(3)
            P(2) = P(4)
            P(3) =REAL( X1(1))
            P(4) =REAL( Y1(1))
            STATUS = PPLINE(2,2,P)
            P(1)=REAL(X1(1))
            P(2) =REAL( Y1(1))
            P(3) =REAL( X1(2))
            P(4) =REAL( Y1(2))
            STATUS = PPLINE(2,2,P)
            P(3) =REAL( X1(3))
            P(4) =REAL( Y1(3))
            STATUS = PPLINE(2,2,P)
            P(3) =REAL(X1(4))
```

```
                    P(4) =REAL(Y1(4))
                    STATUS = PPLINE(2,2,P)
                    P(1)=REAL(X1(2))
                    P(2) =REAL(Y1(2))
                    P(3) =REAL( X1(3))
                    P(4) =REAL( Y1(3))
                    STATUS = PPLINE(2,2,P)
                    P(3) =REAL( X1(4))
                    P(4) =REAL( Y1(4))
                    STATUS = PPLINE(2,2,P)
                    P(1)=REAL(X1(3))
                    P(2) =REAL( Y1(3))
                    P(3) =REAL( X1(4))
                    P(4) =REAL( Y1(4))
                    STATUS =-PPLINE(2,2,P)
                    Z=0
                  ENDIF
                ENDIF
              ENDIF
920         CONTINUE
           Z=0
927      CONTINUE
930    CONTINUE
      READ(*,932) ANS
932    FORMAT(A1)
      MODE=3
      CALL QSMODE(MODE)
      STATUS=PCLIOS(7,DEVIC)
      STATUS=PCLSPS()
935    RETURN
      END
C
C
C
      SUBROUTINE MOLPOT(QEQ,Q,V,MOLNAM,N,MASSES,LORB)
C
C
      INTEGER*2 N
      REAL*8 V,Q(1),QEQ(1),DE1,RE1,A11,GAMMA2,SINH1,SINH2,SINH3,COSH1,
     X       A12,A13,DE2,RE2,A21,A22,A23,DE3,RE3,A31,A32,A33,RR1,RR2,
     X       VR1,VR2,VR3,VDIAT,INT1,INT2,INT3,X,Y,Z,HBAR,C,GAMMA1,C33,
     X       GAMMA3,TANH1,TANH2,TANH3,S1,S2,S3,VIO,C1,C2,C3,C11,C22,
     X       C12,C13,C23,C111,C222,C333,C112,C122,C113,C133,C223,C233,
     X       C123,C1111,C2222,C3333,C1112,C1122,C1222,C1113,C1133,C1333
     X       C2223,C2233,C2333,C1123,C1223,C1233,C22222,V3XYZ,P,PI,RR3,
     X       COSH2,COSH3,MASSES(1)
      CHARACTER MOLNAM*20,LORB*1
C
C     The Murrel, Carter, and Halonen Potential for HCN<—>CNH (Jour-
C     nal of Molecular Spectroscopy 93 p.307 1982) is used. The force
C     constants are such that the potential is in eV. This is first con-
C     verted to aJ. (1 eV = 0.16022E-18 Joules = 0.16022 aJoules.) This
C     potential is converted at the end to cm^-1. The conversion factor
```

```
C     is E(cm.^-1) = (1E-20/2*PI*HBAR*C). All constants are in SI units
C     Equilibrium coordinates found by printing Q's when V at its min-
C     imum.
C
      N=3
      MOLNAM='HCN <--->  CNH       '
      MASSES(1)=1.0
      MASSES(2)=12.0
      MASSES(3)=14.0
      QEQ(1)=1.06549
      QEQ(2)=1.15321
      QEQ(3)=2.2187
C
C     Diatomic part of the potential.
C
      DATA DE1 /2.8521/, DE2 /7.9282/, DE3 /3.9938/
      DATA RE1 /1.0823/, RE2 /1.1718/, RE3 /1.0370/
      DATA A11 /5.5297/, A21 /5.2448/, A31 /3.0704/
      DATA A12 /8.7166/, A22 /7.3416/, A32 /0.0000/
      DATA A13 /5.3082/, A23 /4.9785/, A33 /0.0000/
      RR1=Q(1)-RE1
      RR2=Q(2)-RE2
      RR3=Q(3)-RE3
      VR1=-DE1*(1+A11*RR1+A12*RR1**2+A13*RR1**3)*EXP(-A11*RR1)
      VR2=-DE2*(1+A21*RR2+A22*RR2**2+A23*RR2**3)*EXP(-A21*RR2)
      VR3=-DE3*(1+A31*RR3+A32*RR3**2+A33*RR3**3)*EXP(-A31*RR3)
      VDIAT=VR1+VR2+VR3
C
C     Triatomic part of the potential.
C
      DATA INT1 /1.9607/, INT2 /2.2794/, INT3 /1.8687/
      X=Q(1)-INT1
      Y=Q(2)-INT2
      Z=Q(3)-INT3
      DATA HBAR /1.054591980D-34/, C /2.997925010D8/
      DATA GAMMA1 /3.9742/, GAMMA2 /4.3688/, GAMMA3 /1.5176/
      SINH1=(EXP(GAMMA1*X/2)-EXP(-(GAMMA1*X/2)))/2
      SINH2=(EXP(GAMMA2*Y/2)-EXP(-(GAMMA2*Y/2)))/2
      SINH3=(EXP(GAMMA3*Z/2)-EXP(-(GAMMA3*Z/2)))/2
      COSH1=(EXP(GAMMA1*X/2)+EXP(-(GAMMA1*X/2)))/2
      COSH2=(EXP(GAMMA2*Y/2)+EXP(-(GAMMA2*Y/2)))/2
      COSH3=(EXP(GAMMA3*Z/2)+EXP(-(GAMMA3*Z/2)))/2
      TANH1=SINH1/COSH1
      TANH2=SINH2/COSH2
      TANH3=SINH3/COSH3
      S1=0.4436*X+0.6091*Y+0.6575*Z
      S2=-0.8941*X+0.2498*Y+0.3718*Z
      S3=0.0622*X-0.7527*Y+0.6554*Z
      DATA V10 /-3.0578/
      DATA C1 /1.9076/, C2 /-0.5008/, C3 /-0.0149/
      DATA C11 /0.6695/, C22 /-1.3535/, C33 /-1.0501/, C12 /0.2698/
      DATA C13 /-1.1120/, C23 /1.9310/, C111 /-0.0877/, C222 /0.0044/
      DATA C333 /0.07/, C112 /0.0898/, C122 /-1.0186/, C113 /-0.0911/
```

```
      DATA C133 /0.0017/, C223 /0.4567/, C233 /-0.8840/, C123 /0.3333/
      DATA C1111 /-0.0367/, C2222 /0.4821/, C3333 /0.2564/
      DATA C1112 /-0.0017/, C1122 /-0.2278/, C1222 /-0.1287/
      DATA C1113 /0.1759/, C1133 /-0.0399/, C1333 /-0.1447/
      DATA C2223 /-0.3147/, C2233 /0.1233/, C2333 /0.3161/
      DATA C1123 /0.0919/, C1223 /-0.0954/, C1233 /0.1778/
      DATA C22222 /-0.1892/
      V3XYZ=(1-TANH1)*(1-TANH2)*(1-TANH3)*V10
      P=1+C1*S1+C2*S2+C3*S3+C11*S1**2+C12*S1*S2+C13*S1*S3+C22*S2**2+C2
     X  S2*S3+C33*S3**2+C111*S1**3+C222*S2**3+C333*S3**3+C112*S2*S1**2
     X  C122*S1*S2**2+C113*S3*S1**2+C133*S1*S3**2+C223*S3*S2**2+C233*S
     X  S3**2+C123*S1*S2*S3+C1111*S1**4+C2222*S2**4+C3333*S3**4+C1112*
     X  S2*S1**3+C1122*S1**2*S2**2+C1222*S1*S2**3+C1113*S3*S1**3+C1133
     X  S1**2*S3**2+C1333*S1*S3**3+C2223*S3*S2**3+C2233*S2**2*S3**2+
     X  C2333*S2*S3**3+C1123*S2*S3*S1**2+C1223*S1*S3*S2**2+C1233*S1*S2
     X  S3**2+C22222*S2**5
      V3XYZ=V3XYZ*P
C
C     Total Potential
C
      V=VDIAT+V3XYZ
      V=0.16022*V
      PI=ACOS(-1.0)
      V=V*(1.0D-20/(2*PI*HBAR*C))
      RETURN
      END
C
C
C
      SUBROUTINE TOJAC(Y,N,MAX4,TEMP1,TEMP2)
C
C
      INTEGER*2 N,MAX4
      REAL*8 Y(MAX4,1),TEMP1(MAX4,1),TEMP2(MAX4,1)
C
C     CALCULATE TARGET (E.G. POLAR) COORDINATES FROM BRI COORDINATES.
C
      RETURN
      END
C
C
C
      SUBROUTINE FROMJA(Y,N,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,EXTRA,TRNAME)
C
C
      INTEGER*2 N,MAX4
      REAL*8 Y(MAX4,1),TEMP1(MAX4,1),TEMP2(MAX4,1),TEMP3(MAX4,1),
     X       TEMP4(MAX4,1),EXTRA(MAX4,1)
      CHARACTER TRNAME*20
```

```
C
C
C     CALCULATE BRI COORDINATES FROM TARGET (E.G. POLAR) COORDINATES.
C
      TRNAME='BAS. INVARIANT (CHA)'
      RETURN
      END
C
C
C
      SUBROUTINE SEPTAR(X,Q,QEQ,N,S,F9,VEQ,MAX4,TEMP1,TEMP2,TEMP3,TEMP
     X                  FIR,Y,XFIR,SEC,XSEC,XEQ,MOLNAM,EXTRA,TRNAME,
     X                  MASSES,LORB,ERROR)
C
C
      INTEGER*2 FIR,SEC,MAX4,N
      REAL*8 V1,V2,XFIR(1),Q(1),QEQ(1),V,S(MAX4,1),VEQ,X(1),XSEC(1),F9
     X       TEMP1(MAX4,1),TEMP2(MAX4,1),TEMP3(MAX4,1),TEMP4(MAX4,1),
     X       XEQ(1),EXTRA(MAX4,1),Y(MAX4,1),MASSES(1)
      LOGICAL ERROR
      CHARACTER MOLNAM*20,TRNAME*20,LORB*1
C
C
C     CALCULATE SEPARABLE POTENTIALS FOR EACH COORDINATE.
C
      IF (((FIR .LE. 2) .AND. (N .EQ. 3)) .OR. ((FIR .LE. 3) .AND.
     X    (N .EQ. 4))) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X            TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=V-VEQ
      ENDIF
      IF (((FIR .EQ. 3) .AND. (N .EQ. 3)) .OR. ((FIR .EQ. 4) .AND.
     X    (N .EQ. 4))) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X            TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=(1.0/(X(1)*X(1))+1.0/(X(2)*X(2)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(2)*XEQ(2)))*(V-VEQ)
      ENDIF
      IF ((FIR .EQ. 5) .AND. (N .EQ. 4)) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X            TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=(1.0/(X(1)*X(1))+1.0/(X(3)*X(3)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(3)*XEQ(3)))*(V-VEQ)
      ENDIF
      IF ((((SEC .LE. 2) .AND. (N .EQ. 3)) .OR. ((SEC .LE. 3) .AND.
     X    (N .EQ. 4))) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X            TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=V-VEQ
      ENDIF
```

```
      IF ((((SEC .EQ. 3) .AND. (N .EQ. 3)) .OR. ((SEC .EQ. 4) .AND.
     X     (N .EQ. 4))) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=(1.0/(X(1)*X(1))+1.0/(X(2)*X(2)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(2)*XEQ(2)))*(V-VEQ)
      ENDIF
      IF (((SEC .EQ. 5) .AND. (N .EQ. 4)) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=(1.0/(X(1)*X(1))+1.0/(X(3)*X(3)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(3)*XEQ(3)))*(V-VEQ)
      ENDIF
      IF ((FIR .EQ. 6) .AND. (N .EQ. 4)) THEN
         V1=0.0
      ENDIF
      IF ((SEC .EQ. 6) .AND. (N .EQ. 4)) THEN
         V2=0.0
      ENDIF
      F9=V1+V2
      RETURN
      END
C
C
C
      SUBROUTINE DIFF(X,Q,QEQ,N,S,F9,VEQ,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,
     X                FIR,Y,XFIR,SEC,XSEC,XEQ,MOLNAM,EXTRA,TRNAME,
     X                MASSES,X2,LORB,ERROR)
C
C
      INTEGER*2 FIR,SEC,MAX4,N
      REAL*8 V1,V2,XFIR(1),Q(1),QEQ(1),V,S(MAX4,1),VEQ,X(1),XSEC(1),F9
     X       TEMP1(MAX4,1),TEMP2(MAX4,1),TEMP3(MAX4,1),TEMP4(MAX4,1),
     X       XEQ(1),EXTRA(MAX4,1),Y(MAX4,1),MASSES(1),TOTAL,X2(1)
      LOGICAL ERROR
      CHARACTER MOLNAM*20,TRNAME*20,LORB*1
C
C
C     CALCULATE DIFFERENCE BETWEEN TOTAL AND SEPARABLE POTENTIALS (I.E.
C     NON-SEPARABLE POTENTIALS).
C
      CALL FNV(X2,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X         TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
      TOTAL=V-VEQ
      IF ((((FIR .LE. 2) .AND. (N .EQ. 3)) .OR. ((FIR .LE. 3) .AND.
     X    (N .EQ. 4))) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=V-VEQ
      ENDIF
```

```
      IF ((((FIR .EQ. 3) .AND. (N .EQ. 3)) .OR. ((FIR .EQ. 4) .AND.
     X    (N .EQ. 4))) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=(1.0/(X(1)*X(1))+1.0/(X(2)*X(2)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(2)*XEQ(2)))*(V-VEQ)
      ENDIF
      IF (((FIR .EQ. 5) .AND. (N .EQ. 4)) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=(1.0/(X(1)*X(1))+1.0/(X(3)*X(3)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(3)*XEQ(3)))*(V-VEQ)
      ENDIF
      IF ((((SEC .LE. 2) .AND. (N .EQ. 3)) .OR. ((SEC .LE. 3) .AND.
     X     (N .EQ. 4))) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=V-VEQ
      ENDIF
      IF ((((SEC .EQ. 3) .AND. (N .EQ. 3)) .OR. ((SEC .EQ. 4) .AND.
     X     (N .EQ. 4))) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=(1.0/(X(1)*X(1))+1.0/(X(2)*X(2)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(2)*XEQ(2)))*(V-VEQ)
      ENDIF
      IF (((SEC .EQ. 5) .AND. (N .EQ. 4)) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=(1.0/(X(1)*X(1))+1.0/(X(3)*X(3)))/(1.0/(XEQ(1)*XEQ(1))
     X       +1.0/(XEQ(3)*XEQ(3)))*(V-VEQ)
      ENDIF
      IF (((FIR .EQ. 6) .AND. (N .EQ. 4)) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XFIR,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V1=V-VEQ
      ENDIF
      IF (((SEC .EQ. 6) .AND. (N .EQ. 4)) .AND. (.NOT. ERROR)) THEN
         CALL FNV(XSEC,N,S,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,EXTRA,
     X           TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
         V2=V-VEQ
      ENDIF
      F9=TOTAL-(V1+V2)
      RETURN
      END
C
C
C
      SUBROUTINE QEQXEQ(QEQ,REQ,N,LORB,Y,MAX4,0,TEMP1,TEMP2,TEMP3,TEMP
     X                  EXTRA,TRNAME,X)
C
C
      INTEGER*2 N,MAX4,A
```

```
      REAL*8 QEQ(1),REQ(1),Y(MAX4,1),X(1),TEMP1(MAX4,1),O(MAX4,1),
     X       TEMP2(MAX4,1),TEMP3(MAX4,1),TEMP4(MAX4,1),EXTRA(MAX4,1)
      CHARACTER TRNAME*20,LORB*1
C
C
C     TRANSFORM EQUILIBRIUM BOND DISTANCE-BOND ANGLE COORDINATES TO
C     EQUILIBRIUM TARGET COORDINATES.
C
      CALL FSOURC(QEQ,REQ,N,LORB)
      CALL FORM(REQ,Y,N,MAX4)
      A=1
      CALL TRFORM(Y,N,A,O,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,EXTRA,TRNAME)
      CALL FORMIN(Y,X,N,MAX4)
      IF (N .EQ. 4) THEN
        CALL TOPHI(X)
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE FNV(X,N,O,V,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,MOLNAM,
     X               EXTRA,TRNAME,Y,QEQ,Q,MASSES,LORB,ERROR)
C
C
      INTEGER*2 N,MAX4,SIZE,A,D
      REAL*8 V,O(MAX4,1),X(1),Y(MAX4,1),TEMP1(MAX4,1),TEMP2(MAX4,1),
     X       SCALAR,TEMP3(MAX4,1),TEMP4(MAX4,1),EXTRA(MAX4,1),QEQ(1),
     X       Q(1),MASSES(1)
      LOGICAL ERROR
      CHARACTER MOLNAM*20,TRNAME*20,LORB*1
C
C
C     TRANSFORM CURRENT SET OF TARGET COORDINATES TO BOND DISTANCE
C     COORDINATES AND CALCULATE POTENTIAL VALUE.
C
      D=N-1
      SCALAR=0.0
      CALL MATSCA(TEMP2,SCALAR,TEMP1,D,MAX4)
      CALL ASSIGN(TEMP1,TEMP2,D,MAX4)
      IF (N .EQ. 4) THEN
        CALL FROMPH(X,ERROR)
      ENDIF
      CALL FORM(X,TEMP1,N,MAX4)
      A=2
      CALL TRFORM(TEMP1,N,A,O,MAX4,TEMP2,TEMP3,TEMP4,Y,EXTRA,TRNAME)
      CALL FORMIN(TEMP1,X,N,MAX4)
      CALL TSOURC(X,Q,N,LORB,ERROR)
      IF (.NOT. ERROR) THEN
        CALL MOLPOT(QEQ,Q,V,MOLNAM,N,MASSES,LORB)
      ENDIF
      RETURN
      END
```

Appendix C

Fortran–77 Source Code for *NUMROV*

```
$NOFLOATCALLS
$STORAGE:2
$LARGE
C
C
C
      PROGRAM MAIN
C
C
C     THIS PROGRAM SOLVES THE FOLLOWING EQUATION:
C
C
C     [A(x)d^2/dx^2 + B(x)d/dx] ψ(x) = [V(x)–E] ψ(x)
C
C
      INTEGER*2 ALG,ECOUNT,LBC,ASIZE,MAX1,ESIZE,POINTS,PSILOW,PSIUPP,
     X        SQ,UBC
      REAL*8 A(5001),B(5001),V(5001),XMIN,XMAX,H,EMIN,EMAX,EIVAL(70,2),
     X      YMAT(5001),ZMAT(5001),U(5001),R(5001),S(5001),T(5001),
     X      FUNCT(5001),POTA1,MINVAL
      CHARACTER ENAME*20,FNAME*20,ANAME*18,BNAME*18,VNAME*20
      LOGICAL ERROR
C
C
      ASIZE=5001
      ESIZE=70
5     CALL WRITEL(25)
      WRITE(*,*) 'GIVE THE NUMBER OF GRID POINTS.'
```

```
      READ(*,*) POINTS
      CALL WRITEL(5)
      IF (POINTS .GT. ASIZE) THEN
        WRITE(*,*) '*** ERROR ***'
        WRITE(*,*)
        MAX1=ASIZE+1
        WRITE(*,10) MAX1
10     FORMAT(' POINTS MUST BE LESS THAN ',I4,'.')
        CALL WRITEL(10)
        CALL HOLD
        GOTO 5
      ENDIF
      WRITE(*,*) 'GIVE XMIN.'
      READ(*,*) XMIN
      CALL WRITEL(5)
      WRITE(*,*) 'GIVE XMAX.'
      READ(*,*) XMAX
      H=(XMAX-XMIN)/(POINTS-1)
      CALL WRITEL(25)
C
C
      CALL ABCALC(A,B,POINTS,H,XMIN,ANAME,BNAME)
      CALL VCALC(V,POINTS,H,XMIN,SQ,POTA1,ENAME,VNAME)
C
C
      CALL SETUP(A,B,ALG,EMIN,EMAX,FNAME,LBC,UBC,POINTS,PSILOW,
     X        PSIUPP,XMIN,XMAX)
      CALL NUMROV(EMIN,EMAX,XMIN,XMAX,POINTS,A,B,V,LBC,UBC,PSILOW,
     X         PSIUPP,SQ,EIVAL,ESIZE,ECOUNT,U,T,R,S,YMAT,ZMAT,FUNCT,
     X         ALG,FNAME,ERROR,POTA1,MINVAL,H)
      CALL EVIEW(EIVAL,ESIZE,ENAME,ECOUNT,ERROR,POINTS,XMIN,XMAX,ANAME,
     X        BNAME,VNAME,EMIN,EMAX,LBC,UBC)
      STOP
      END
C
C
C
      SUBROUTINE SETUP(A,B,ALG,EMIN,EMAX,FNAME,LBC,UBC,POINTS,PSILOW,
     X            PSIUPP,XMIN,XMAX)
C
C
C     THIS SUBROUTINE PROMPTS THE USER FOR THE UPPER BOUNDARY CON-
C     DITION, THE LOWER BOUNDARY CONDITION, AND THE MINIMUM AND MAXIMUM
C     EIGENVALUE LIMITS.
C
      INTEGER*2 ALG,LBC,UBC,POINTS,PSILOW,PSIUPP,I
      REAL*8 A(1),B(1),EMIN,EMAX,XMIN,XMAX
      CHARACTER FNAME*20
      LOGICAL ERROR
C
C
C
```

```
      ERROR=.FALSE.
      ALG=2
      DO 5 I=1,POINTS
        IF ((ABS(B(I))) .GT. 0.0) THEN
          ALG=1
          GOTO 10
        ENDIF
5     CONTINUE
10    CALL WRITEL(20)
20    WRITE(*,*) '       CHOOSE UPPER AND LOWER BOUNDARY CONDITIONS FR
     XOM:'
      CALL WRITEL(2)
      WRITE(*,25) XMIN,XMAX
25    FORMAT('       1.) PSI(',F10.3,') = 0.0    OR      PSI(',F10
     X.3,') = 0.0')
      WRITE(*,30) XMIN,XMAX
30    FORMAT('       2.) dPSI/dX(',F10.3,') = 0.0   OR    dPSI/dX('
     X,F10.3,') = 0.0')
      WRITE(*,35)
35    FORMAT('       3.) FROBENIUS EXPANSION OF PSI AT AND NEAR THE B
     XOUND.')
      WRITE(*,36) XMIN,XMAX
36    FORMAT('       4.) PSI(',F10.3,') = PSI(',F10.3,')   AND')
      WRITE(*,37) XMIN,XMAX
37    FORMAT('          dPSI/dX(',F10.3,') = dPSI/dX(',F10.3,') = 0.
     XO')
      WRITE(*,38) XMIN,XMAX
38    FORMAT('       5.) PSI(',F10.3,') = PSI(',F10.3,') = 0.0   AND
     X')
      WRITE(*,39) XMIN,XMAX
39    FORMAT('          dPSI/dX(',F10.3,') = dPSI/dX(',F10.3,')')
      CALL WRITEL(10)
      IF (.NOT. ERROR) THEN
        WRITE(*,*) 'A.) GIVE THE LOWER BOUND CONDITION.'
        READ(*,*) LBC
        CALL WRITEL(3)
        IF ((LBC .EQ. 3) .AND. (ALG .EQ. 2)) THEN
          WRITE(*,*) '*** ERROR ***'
          WRITE(*,*)
          WRITE(*,40)
40         FORMAT(' THE FROBENIUS EXPANSION AT THE LOWER BOUND IS FOR L
     XEGENDRE TYPE EQUATIONS ONLY.')
          CALL WRITEL(10)
          CALL HOLD
          CALL WRITEL(25)
          GOTO 20
        ENDIF
        IF ((LBC .GE. 6) .OR. (LBC .EQ. 0)) THEN
          WRITE(*,*) '*** ERROR ***'
          WRITE(*,*)
          WRITE(*,45)
45         FORMAT(' YOU CHOSE A BOUNDARY CONDITION THAT DOES NOT EXIST.
     X')
```

```
        CALL WRITEL(10)
        CALL HOLD
        CALL WRITEL(25)
        GOTO 20
      ENDIF
    ENDIF
    WRITE(*,*) 'B.) GIVE THE UPPER BOUND CONDITION.'
    READ(*,*) UBC
    CALL WRITEL(5)
    IF ((UBC .EQ. 3) .AND. (ALG .EQ. 2)) THEN
      ERROR=.TRUE.
      WRITE(*,*) '*** ERROR ***'
      WRITE(*,*)
      WRITE(*,50)
50     FORMAT(' THE FROBENIUS EXPANSION AT THE UPPER BOUND IS FOR LEGE
   XNDRE TYPE EQUATIONS ONLY.')
      CALL WRITEL(10)
      CALL HOLD
      CALL WRITEL(25)
      GOTO 20
    ENDIF
    IF ((UBC .GE. 6) .OR. (UBC .EQ. 0)) THEN
      ERROR=.TRUE.
      WRITE(*,*) '*** ERROR ***'
      WRITE(*,*)
      WRITE(*,55)
55     FORMAT(' YOU CHOSE A BOUNDARY CONDITION THAT DOES NOT EXIST.')
      CALL WRITEL(10)
      CALL HOLD
      CALL WRITEL(25)
      GOTO 20
    ENDIF
    IF (((LBC .EQ. 4) .OR. (UBC .EQ. 4)) .AND. ((UBC .NE. LBC) .AND.
   X    ((UBC .NE. 5) .AND. (LBC .NE. 5)))) THEN
      WRITE(*,*) '*** ERROR ***'
      WRITE(*,*)
      WRITE(*,*) 'BOUNDARY CONDITION 4.) APPLIES TO BOTH BOUNDS SIMUL
   XTANEOUSLY !!'
      CALL WRITEL(10)
      CALL HOLD
      CALL WRITEL(25)
      GOTO 20
    ENDIF
    IF (((LBC .EQ. 5) .OR. (UBC .EQ. 5)) .AND. ((UBC .NE. LBC) .AND.
   X    ((UBC .NE. 4) .AND. (LBC .NE. 4)))) THEN
      WRITE(*,*) '*** ERROR ***'
      WRITE(*,*)
      WRITE(*,*) 'BOUNDARY CONDITION 5.) APPLIES TO BOTH BOUNDS SIMUL
   XTANEOUSLY !!'
      CALL WRITEL(10)
      CALL HOLD
      CALL WRITEL(25)
      GOTO 20
```

```
      ENDIF
      IF (((UBC .EQ. 4) .OR. (UBC .EQ. 5)) .AND. ((LBC .NE. UBC) .AND.
     X    ((LBC .EQ. 4) .OR. (LBC .EQ. 5)))) THEN
        WRITE(*,*) '*** ERROR ***'
        WRITE(*,*)
        WRITE(*,*) 'YOU CANNOT CHOOSE CONDITION 4.) FOR ONE BOUND AND C
     XONDITION 5.) FOR THE OTHER.'
        CALL WRITEL(10)
        CALL HOLD
        CALL WRITEL(25)
        GOTO 20
      ENDIF
      PSILOW=3
      PSIUPP=3
60    IF (LBC .EQ. 3) THEN
        WRITE(*,*) 'GIVE THE NUMBER OF VALUES OF PSI (AT THE LOWER BOUN
     XD) TO BE CALCULATED BY THE'
        WRITE(*,*) 'FROBENIUS EXPANSION. (MUST BE GREATER THAN 1.)'
        READ(*,*) PSILOW
        CALL WRITEL(5)
      ENDIF
      IF (PSILOW .LT. 2) THEN
        WRITE(*,*) '*** ERROR ***'
        WRITE(*,*)
        WRITE(*,*) 'AT LEAST TWO VALUES OF PSI AT THE LOWER BOUND MUST
     XBE CALCULATED BY THE'
        WRITE(*,*) 'FROBENIUS EXPANSION.'
        CALL WRITEL(10)
        CALL HOLD
        CALL WRITEL(25)
        GOTO 60
      ENDIF
65    IF (UBC .EQ. 3) THEN
        WRITE(*,*) 'GIVE THE NUMBER OF VALUES OF PSI (AT THE UPPER BOUN
     XD) TO BE CALCULATED BY THE'
        WRITE(*,*) 'FROBENIUS EXPANSION. (MUST BE GREATER THAN 1.)'
        READ(*,*) PSIUPP
        CALL WRITEL(5)
      ENDIF
      IF (PSIUPP .LT. 2) THEN
        WRITE(*,*) '*** ERROR ***'
        WRITE(*,*)
        WRITE(*,*) 'AT LEAST TWO VALUES OF PSI AT THE UPPER BOUND MUST
     XBE CALCULATED BY THE'
        WRITE(*,*) 'FROBENIUS EXPANSION.'
        CALL WRITEL(10)
        CALL HOLD
        CALL WRITEL(25)
        GOTO 65
      ENDIF
      CALL WRITEL(20)
      WRITE(*,*) 'GIVE MIN. LIMIT FOR EIGENVALUES.'
      READ(*,*) EMIN
```

```
      CALL WRITEL(5)
      WRITE(*,*) 'GIVE MAX. LIMIT FOR EIGENVALUES.'
      READ(*,*) EMAX
      CALL WRITEL(3)
      WRITE(*,*) 'GIVE THE NAME OF THE FILE IN WHICH THE EIGENFUNCTIONS
     XWILL BE STORED.'
      READ(*,70) FNAME
70    FORMAT(A20)
      RETURN
      END
C
C
C
      SUBROUTINE NUMROV(EMIN,EMAX,XMIN,XMAX,POINTS,A,B,V,LBC,UBC,PSILOW,
     X            PSIUPP,SQ,EIVAL,ESIZE,ECOUNT,U,T,R,S,YMAT,ZMAT,
     X            FUNCT,ALG,FNAME,ERROR,POTA1,MINVAL,H)
C
C
C     THIS SUBROUTINE CALCULATES A SET OF EIGENVALUES AND EIGENFUNCTIONS
C     FOR A GIVEN D.E. THE EIGENFUNCTIONS AND EIGENVALUES ARE SAVED TO
C     DISK.
C
C
C     ACCEPTS: EMIN--THE MINIMUM LIMIT FOR THE EIGENVALUES.
C           EMAX--THE MAXIMUM LIMIT FOR THE EIGENVALUES.
C           XMIN--X AT LOWER BOUND.
C           XMAX--X AT UPPER BOUND.
C           POINTS--THE NUMBER OF GRID POINTS.
C           A--FUNCTION IN FRONT OF THE SECOND DERIVATIVE TERM.
C           B--FUNCTION IN FRONT OF THE FIRST DERIVATIVE TERM.
C           V--THE POTENTIAL FUNCTION.
C           LBC--THE LOWER BOUNDARY CONDITION. (PSI=0 , ETC.)
C           UBC--THE UPPER BOUNDARY CONDITION. (PSI=0 , ETC.)
C           SQ--THE ANGULAR MOMENTUM QUANTUM NUMBER USED IN SOME OF
C             THE VCALC SUBROUTINES.
C           PSILOW--THE NUMBER OF VALUES OF PSI CALCULATED BY THE FRO-
C                BENIUS EXPANSION AT THE LOWER BOUND.
C           PSIUPP--THE NUMBER OF VALUES OF PSI CALCULATED BY THE FRO-
C                BENIUS EXPANSION AT THE UPPER BOUND.
C           EIVAL--MEMORY SPACE FOR THE EIGENVALUES AND QUANTUM NUM-
C               BERS.
C           ESIZE--THE NUMBER OF ROWS OF EIVAL.
C           ECOUNT--MEMORY SPACE FOR THE NUMBER OF EIGENVALUES CALCU-
C                LATED BETWEEN EMIN AND EMAX.
C           T--THE FUNCTION T, USED TO CALCULATE U.
C           U--THE FUNCTION U IN THE TWO-TERM RECURSION RELATIONS.
C           R--THE FUNCTION R IN ONE OF THE TWO-TERM RECURSION RELA-
C             TIONS.
C           S--THE FUNCTION S IN ONE OF THE TWO-TERM RECURSION RELA-
C             TIONS.
C           YMAT--THE FUNCTION Y IN THE TWO-TERM RECURSION RELATIONS.
C           ZMAT--THE FUNCTION Z IN THE TWO-TERM RECURSION RELATIONS.
C           FUNCT--THE WAVE-FUNCTION PSI.
```

```
C          ALG--INDICATES THE ALGORITHM USED.
C          FNAME--THE NAME OF THE FILE TO WHICH THE EIGENVALUES AND
C              EIGENFUNCTIONS ARE WRITTEN.
C          ERROR--ASSIGNED THE VALUE .TRUE. IF NO EIGENVALUES AND
C              EIGENFUNCTIONS WERE CALCULATED.
C          POTA1--THE COEFFICIENT OF THE LINEAR TERM IN THE TAYLOR
C              EXPANSION OF THE POTENTIAL. (USED IN THE FROBENIUS
C              EXPANSION.)
C          MINVAL--THE MINIMUM VALUE OF THE POTENTIAL, V(I).
C          H--DEFINED AS (XMAX--XMIN)/(POINTS--1).
C
C
C    RETURNS: ECOUNT--THE NUMBER OF EIGENVALUES CALCULATED BETWEEN EMIN
C               AND EMAX, INCLUSIVE.
C          EIVAL--THE EIGENVALUES AND QUANTUM NUMBERS CALCULATED.
C          ERROR--(A FLAG)--ASSIGNED THE VALUE .TRUE. IF NO EIGENVALUES
C              AND EIGENFUNCTIONS WERE CALCULATED.
C
C
     INTEGER*2 POINTS,LBC,UBC,PSILOW,PSIUPP,SQ,ESIZE,ECOUNT,ALG,
    X       BTYPE,COUNT,CYCLE,DISK,EVAL,I,ITER,K,LASTQN,MAX,
    X       MAX1,QN
     REAL*8 EMIN,EMAX,XMIN,XMAX,A(1),B(1),V(1),EIVAL(ESIZE,1),U(1),
    X     R(1),S(1),YMAT(1),ZMAT(1),FUNCT(1),POTA1,AC,DE,DK1,DK2,
    X     E(20),EI,EPREV,GUESS,GUESS2,H,T1,T2,MINVAL,T(1)
     LOGICAL ERROR,FIRST2,FOUND,LOOP,WAY1,USUAL,BYTW01,BYTW02
     CHARACTER FNAME
C
C
     CALL INITZE(A,B,AC,ALG,BTYPE,BYTW01,BYTW02,CYCLE,DE,EMAX,FOUND,
    X        FIRST2,GUESS,H,ITER,LBC,UBC,MINVAL,POINTS,USUAL,V,
    X        WAY1,YMAT,ZMAT)
     ITER=20
     CALL WRITEL(25)
     WRITE(*,*) 'CALCULATING EIGENVALUES AND EIGENFUNCTIONS ...'
     CALL WRITEL(3)
C
C    THE PRINCIPLE LOOP IN NUMROV. GUESS IS ASSIGNED AN INITIAL VALUE
C    AND THE NEXT EIGENVALUE IN LINE IS SEARCHED FOR. THE PROGRAM MAY
C    PASS THROUGH THIS LOOP A NUMBER OF TIMES BEFORE THE NEXT EIGEN--
C    VALUE IS FOUND.
C
25   GUESS=GUESS--EI
C
C      NEXT EIGENVALUE HAS ALREADY BEEN CALCULATED IF FOUND IS TRUE.
C      GO TO 48 TO CALCULATE FUNCTIONS R AND S FOR THE FOUND EIGEN--
C      VALUE.
C
       IF ((FOUND) .AND. (COUNT .EQ. 0)) THEN
         K=1
```

```
        IF (.NOT. USUAL) THEN
          QN=LASTQN-2
        ELSE
          QN=LASTQN-1
        ENDIF
        E(K)=GUESS2
        GOTO 48
      ENDIF
      LOOP=.FALSE.
      K=1
      E(K)=GUESS
      write(*,34) K,E(K)
34     format(' ESTIMATE(',I2,') IS:',F25.10)
      CALL NUMERO(POINTS,E,V,DK1,DK2,U,R,S,T,K,QN,H,
     X          PSILOW,PSIUPP,XMIN,XMAX,SQ,YMAT,ZMAT,ALG,BTYPE,
     X          A,EVAL,POTA1)
C
C     NEW ESTIMATES FOR THE EIGENVALUES ARE CALCULATED IN TWO DISTINCT
C     WAYS. WAY1 IS NEWTON'S METHOD. THE OTHER WAY (NOT WAY1) IS THE
C     BISECTION METHOD.
C
      IF (.NOT. WAY1) THEN
        GOTO 55
      ENDIF
      K=K+1
      E(K)=E(K-1)+DE
      write(*,35) K,E(K)
35     format(' ESTIMATE(',I2,') IS:',F25.10)
      CALL NUMERO(POINTS,E,V,DK1,DK2,U,R,S,T,K,QN,H,
     X          PSILOW,PSIUPP,XMIN,XMAX,SQ,YMAT,ZMAT,ALG,BTYPE,
     X          A,EVAL,POTA1)
45     K=K+1
       T1=ABS(E(K-1)-E(K-2))
       T2=ABS(DK1-DK2)
C
C      NEWTON'S METHOD.
C
       IF (T2 .NE. 0.0) THEN
         E(K)=E(K-1)-T1*DK1/T2
         write(*,46) K,E(K)
46        format(' ESTIMATE(',I2,') IS:',F25.10)
       ELSE
         E(K)=E(K-1)+DE
         write(*,47) K,E(K)
47        format(' ESTIMATE(',I2,') IS:',F25.10)
       ENDIF
48     CALL NUMERO(POINTS,E,V,DK1,DK2,U,R,S,T,K,QN,H,
     X          PSILOW,PSIUPP,XMIN,XMAX,SQ,YMAT,ZMAT,ALG,
     X          BTYPE,A,EVAL,POTA1)
```

```
        IF (K .EQ. ITER) THEN
          WRITE(*,50) ITER
50         FORMAT(' CONVERGENCE IS DOUBTFUL ... ',I2,' ITERATIONS HA
     XVE BEEN DONE.')
          GOTO 52
          CALL WRITEL(3)
        ENDIF
        IF ((FOUND) .AND. (COUNT .EQ. 0)) THEN
          FOUND=.FALSE.
          GOTO 52
        ENDIF
      IF (ABS(E(K)-E(K-1)) .GE. AC) GOTO 45
52     IF (CYCLE .EQ. 1) THEN
        CALL CYCLE1(E,EI,QN,LASTQN,K)
      ENDIF
55     IF ((CYCLE .GE. 2) .AND. (USUAL)) THEN
        CALL CYCLE3(E,EI,GUESS,QN,LASTQN,K,EPREV,LOOP,COUNT,POINTS,
     X            FUNCT,R,S,T,H,ALG,EVAL,WAY1,FOUND,GUESS2,MINVAL,
     X            A)
      ELSE
        IF (CYCLE .GE. 2) THEN
          CALL CYCLE4(E,EI,GUESS,QN,LASTQN,K,EPREV,LOOP,COUNT,
     X              POINTS,FUNCT,R,S,T,H,ALG,EVAL,WAY1,FOUND,
     X              GUESS2,MINVAL,A)
        ENDIF
      ENDIF
C
C     NEXT EIGENVALUE NOT FOUND IF LOOP IS TRUE.
C
      IF (LOOP) THEN
        COUNT=COUNT+1
        GOTO 25
      ENDIF
      CYCLE=CYCLE+1
      COUNT=0
C
C     CALCULATE EIGENFUNCTION AND SAVE IT, IF EIGENVALUE IS BETWEEN
C     CHOSEN LIMITS.
C
70     IF (((E(K) .GE. EMIN) .AND. (E(K) .LE. EMAX)) .AND. ((USUAL)
     X      .OR. (((BYTW01) .AND. (MOD(QN,2) .NE. 0)) .OR. ((BYTW02)
     X      .AND. (MOD(QN,2) .EQ. 0))))) THEN
        ECOUNT=ECOUNT+1
        IF (ECOUNT .LE. ESIZE) THEN
          write(*,*) qn,'    ',e(k)
          IF (FIRST2) THEN
            CALL STAT4(FUNCT,POINTS,DISK,FNAME,XMIN,H)
          ENDIF
          CALL EIVECS(FUNCT,R,S,T,H,ALG,POINTS,EVAL)
          CALL STAT3(FIRST2,FUNCT,POINTS,DISK,E,K,FNAME,QN)
          FIRST2=.FALSE.
          LASTQN=QN
          EPREV=E(K)
```

```
            EIVAL(ECOUNT,1)=QN
            EIVAL(ECOUNT,2)=E(K)
            IF ((((USUAL) .OR. (BYTWO2)) .AND. (QN .NE. 0)) .OR.
     X           ((BYTWO1) .AND. (QN .NE. 1))) THEN
     GOTO 25
            ENDIF
          ELSE
            ENDFILE DISK
            CLOSE(DISK,STATUS='KEEP')
            WRITE(*,130) E(K),QN
  130        FORMAT(' THERE IS NO STORAGE SPACE LEFT FOR THE EIGENVALU
     XE',F14.4,' WITH QUANTUM NUMBER ',I2,' .')
            ECOUNT=ECOUNT-1
            CALL WRITEL(10)
            CALL HOLD
          ENDIF
        ELSE
C
C         ASSIGN LASTQN AND EPREV AND RETURN TO CALCULATE MORE EIGEN-
C         VALUES IF LAST ONE HAS NOT BEEN CALCULATED.
C
          IF ((E(K) .GE. EMIN) .AND. ((((USUAL) .OR. (BYTWO2)) .AND.
     X       (QN .NE. 0)) .OR. ((BYTWO1) .AND. (QN .GT. 1)))) THEN
            IF (USUAL) THEN
              LASTQN=QN
            ELSE
              IF ((BYTWO1) .AND. (MOD(QN,2) .NE. 0)) THEN
                LASTQN=QN
              ELSE
                IF ((BYTWO2) .AND. (MOD(QN,2) .EQ. 0)) THEN
                  LASTQN=QN
                ELSE
                  LASTQN=QN+1
                ENDIF
              ENDIF
            ENDIF
            EPREV=E(K)
     GOTO 25
          ELSE
            IF (.NOT. FIRST2) THEN
              ENDFILE DISK
              CLOSE(DISK,STATUS='KEEP')
            ELSE
              ERROR=.TRUE.
              RETURN
            ENDIF
          ENDIF
        ENDIF
      RETURN
      END
```

```
C
C
C
      SUBROUTINE INITZE(A,B,AC,ALG,BTYPE,BYTWO1,BYTWO2,CYCLE,DE,EMAX,
     X                  FOUND,FIRST2,GUESS,H,ITER,LBC,UBC,MINVAL,POINTS,
     X                  USUAL,V,WAY1,YMAT,ZMAT)
C
C
C     THIS SUBROUTINE INITIALIZES A NUMBER OF VARIABLES FOR NUMROV.
C
C
      INTEGER*2 ALG,BTYPE,CYCLE,I,ITER,LBC,UBC,POINTS
      REAL*8 A(1),B(1),AC,DE,EMAX,GUESS,H,MINVAL,V(1),YMAT(1),ZMAT(1)
      LOGICAL BYTWO1,BYTWO2,FIRST2,FOUND,USUAL,WAY1
C
C
      MINVAL=V(1)
C
C     CALCULATE MINIMUM VALUE OF POTENTIAL.
C
      DO 12 I=2,POINTS
        MINVAL=DMIN1(V(I),MINVAL)
12    CONTINUE
C
C     CALCULATE Y AND Z FUNCTIONS.
C
17    IF (ALG .EQ. 1) THEN
        DO 18 I=1,POINTS
          YMAT(I)=1.0-0.5*H*(-B(I)/A(I))
          ZMAT(I)=1.0+0.5*H*(-B(I)/A(I))
18      CONTINUE
      ELSE
        DO 19 I=1,POINTS
          YMAT(I)=1.0
          ZMAT(I)=1.0
19      CONTINUE
      ENDIF
      WAY1=.TRUE.
      USUAL=.TRUE.
      BYTWO1=.FALSE.
      BYTWO2=.FALSE.
      FOUND=.FALSE.
      FIRST2=.TRUE.
      AC=1.0D-07
      CYCLE=1
      IF ((LBC .EQ. 1) .AND. (UBC .EQ. 1)) THEN
        BTYPE=1
      ENDIF
      IF ((LBC .EQ. 2) .AND. (UBC .EQ. 1)) THEN
        BTYPE=2
      ENDIF
```

```
      IF ((LBC .EQ. 1) .AND. (UBC .EQ. 2)) THEN
        BTYPE=3
      ENDIF
      IF ((LBC .EQ. 2) .AND. (UBC .EQ. 2)) THEN
        BTYPE=4
      ENDIF
      IF ((LBC .EQ. 3) .AND. (UBC .EQ. 1)) THEN
        BTYPE=5
      ENDIF
      IF ((LBC .EQ. 3) .AND. (UBC .EQ. 2)) THEN
        BTYPE=6
      ENDIF
      IF ((LBC .EQ. 3) .AND. (UBC .EQ. 3)) THEN
        BTYPE=7
      ENDIF
      IF ((LBC .EQ. 1) .AND. (UBC .EQ. 3)) THEN
        BTYPE=8
      ENDIF
      IF ((LBC .EQ. 2) .AND. (UBC .EQ. 3)) THEN
        BTYPE=9
      ENDIF
      IF (LBC .EQ. 4) THEN
        BTYPE=4
        BYTWO2=.TRUE.
        USUAL=.FALSE.
      ENDIF
      IF (LBC .EQ. 5) THEN
        BTYPE=1
        BYTWO1=.TRUE.
        USUAL=.FALSE.
      ENDIF
      IF (EMAX .NE. 0.0) THEN
        DE=EMAX/1000000.0
        GUESS=EMAX-0.02*EMAX
      ELSE
        DE=0.0002
        GUESS=0.01
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE NUMERO(POINTS,E,V,DK1,DK2,U,R,S,T,K,QN,H,
     X                  PSILOW,PSIUPP,XMIN,XMAX,SQ,YMAT,ZMAT,ALG,BTYPE,
     X                  A,EVAL,POTA1)
C
C
C     THIS SUBROUTINE CALCULATES THE FUNCTIONS R AND S. IT ALSO CAL-
C     CULATES THE MATCHING POINT, EVAL, FOR ANY GIVEN D.E.. IT THEN CAL-
C     CULATES DK1 AT THE MATCHING POINT, AND ASSIGNS TO DK2 THE PREVIOUS
C     VALUE OF DK1.
```

```
C
C
      INTEGER*2 ALG,BTYPE,EVAL,I,J,K,POINTS,PSILOW,PSIUPP,QN,SQ
      REAL*8 E(1),V(1),DK1,DK2,U(1),R(1),S(1),T(1),H,
     X     XMIN,XMAX,YMAT(1),ZMAT(1),QVAL,A(1),POTA1
C
C
C     CALCULATE THE FUNCTION T.
C
      DO 5 I=1,POINTS
        QVAL=(1.0/A(I))*(E(K)-V(I))
        T(I)=-(H*H/12.0)*QVAL
5     CONTINUE
C
C     CALCULATE THE FUNCTION U FOR B(x) NON-ZERO.
C
      IF (ALG .EQ. 1) THEN
        DO 10 I=1,POINTS
          U(I)=2.0+12.0*T(I)
10      CONTINUE
      ENDIF
C
C     CALCULATE THE FUNCTION U FOR B(x) ZERO.
C
      IF (ALG .EQ. 2) THEN
        DO 20 I=1,POINTS
          U(I)=(2.0+10.0*T(I))/(1.0-T(I))
20      CONTINUE
      ENDIF
      CALL BOUCAL(ALG,BTYPE,R,S,U,POINTS,PSILOW,PSIUPP,XMIN,XMAX,SQ,H,E,
     X          K,YMAT,ZMAT,POTA1)
C
C     CALCULATE THE FUNCTION R.
C
      DO 25 I=PSILOW,(POINTS-1)
        R(I)=U(I)/YMAT(I)-ZMAT(I)/YMAT(I)*(1.0/R(I-1))
25    CONTINUE
C
C     CALCULATE THE FUNCTION S.
C
      DO 30 I=(POINTS-(PSIUPP-1)),2,-1
        S(I)=U(I)/ZMAT(I)-YMAT(I)/ZMAT(I)*(1.0/S(I+1))
30    CONTINUE
C
C     CALCULATE THE MATCHING POINT.
C
      DO 40 I=POINTS,2,-1
        IF (S(I) .LE. 1.0) THEN
          J=I
          GOTO 50
        ENDIF
40    CONTINUE
```

```
C
C    ASSIGN THE MATCHING POINT.
C
50    IF (J .EQ. 0) THEN
        EVAL=EVAL+1
      ELSE
        EVAL=J
      ENDIF
      I=2
      J=0
C
C    COUNT THE NUMBER OF NODES.
C
55    IF (R(I) .LT. 0.0) THEN
        J=J+1
        I=I+1
        IF (I .LT. EVAL) GOTO 55
      ELSE
        I=I+1
        IF (I .LT. EVAL) GOTO 55
      ENDIF
      IF (EVAL .EQ. POINTS) THEN
        EVAL=EVAL-3
      ENDIF
      IF (EVAL .EQ. (POINTS-1)) THEN
        EVAL=EVAL-2
      ENDIF
      IF (EVAL .EQ. (POINTS-2)) THEN
        EVAL=EVAL-1
      ENDIF
      IF (EVAL .EQ. 1) THEN
        EVAL=EVAL+3
      ENDIF
      IF (EVAL .EQ. 2) THEN
        EVAL=EVAL+2
      ENDIF
      IF (EVAL .EQ. 3) THEN
        EVAL=EVAL+1
      ENDIF
      DK2=DK1
      DK1=1.0/S(EVAL+1)-R(EVAL)
      QN=J
      RETURN
      END
C
C
C
      SUBROUTINE BOUCAL(ALG,BTYPE,R,S,U,POINTS,PSILOW,PSIUPP,XMIN,
     X                  XMAX,SQ,H,E,K,YMAT,ZMAT,POTA1)
```

```
C
C
C    THIS SUBROUTINE CALCULATES THE VALUES OF R AND S NEAR THE BOUNDS,
C    ACCORDING TO THE USER'S CHOICE OF BOUNDARY CONDITIONS.
C
C
      INTEGER*2 ALG,BTYPE,HIGH,I,J,K,POINTS,PSILOW,PSIUPP,SQ
      REAL*8 R(1),S(1),U(1),PSI(10),CHI(10),F(10),C,A1,A2,E(1),
     X     H,X,CHIDP(10),XMIN,XMAX,YMAT(1),ZMAT(1),AA,BB,CC,POTA1
C
C
      HIGH=10
C
C    F OR PSI IS 0 AT BOTH BOUNDS.
C
      IF (BTYPE .EQ. 1) THEN
        R(1)=1.67D+200
        R(2)=U(2)/YMAT(2)
        S(POINTS-1)=U(POINTS-1)/ZMAT(POINTS-1)
        S(POINTS)=1.67D+200
      ENDIF
C
C    ALGORITHM IS IN F (PROPORTIONAL TO PSI) AND DERIVATIVE
C    OF PSI IS 0 AT LOWER BOUND.
C
      IF (BTYPE .EQ. 2) THEN
        R(1)=U(1)/2.0
        R(2)=U(2)/YMAT(2)-(ZMAT(2)/YMAT(2))*(1.0/R(1))
        S(POINTS-1)=U(POINTS-1)/ZMAT(POINTS-1)
        S(POINTS)=1.67D+200
      ENDIF
C
C    PSI IS ZERO AT LOWER BOUND AND DERIVATIVE OF PSI IS ZERO AT UPPER
C    BOUND.
C
      IF (BTYPE .EQ. 3) THEN
        R(1)=1.67D+200
        R(2)=U(2)/YMAT(2)
        S(POINTS)=U(POINTS)/2.0
        S(POINTS-1)=U(POINTS-1)/ZMAT(POINTS-1)-(YMAT(POINTS-1)/
     X          ZMAT(POINTS-1))*(1.0/S(POINTS))
      ENDIF
C
C    DERIVATIVE OF PSI IS 0 AT BOTH UPPER AND LOWER BOUND.
C
      IF (BTYPE .EQ. 4) THEN
        R(1)=U(1)/2.0
        R(2)=U(2)/YMAT(2)-(ZMAT(2)/YMAT(2))*(1.0/R(1))
        S(POINTS)=U(POINTS)/2.0
        S(POINTS-1)=U(POINTS-1)/ZMAT(POINTS-1)-(YMAT(POINTS-1)/
     X          ZMAT(POINTS-1))*(1.0/S(POINTS))
      ENDIF
```

```
C
C    THE FROBENIUS EXPANSION IN PSI AT ONE OR BOTH BOUNDS WAS CHOSEN.
C    (FOR LEGENDRE TYPE EQUATIONS.)
C
     IF (BTYPE .GE. 5) THEN
       C=SQ/2.0
       A1=(SQ*SQ-SQ-2*E(K))/(4.0*SQ+4.0)
       A2=(SQ**4+5.0*SQ**3-4*SQ*SQ*(E(K)-2.0)-4.0*SQ*(2.0*E(K)-2.0*
    X     POTA1-1.0)+4*(E(K)**2-2.0*E(K)+2.0*POTA1))/(32.0*(SQ+1.0)*
    X     (SQ+2.0))
C
C      Calculate the first PSILOW values of the wave function PSI.
C
       DO 5 I=1,PSILOW
         X=H*(I-1)+XMIN
         PSI(I)=(1.0+X)**C+A1*(1.0+X)**(C+1)+A2*(1.0+X)**(C+2)
5      CONTINUE
C
C      Calculate the first (PSILOW-1) values of R.
C
       DO 10 I=1,(PSILOW-1)
         R(I)=PSI(I+1)/PSI(I)
10     CONTINUE
C
C      Calculate the last PSIUPP values of the wave function PSI.
C
       J=HIGH
       DO 15 I=POINTS,(POINTS-PSIUPP+1),-1
         X=H*(I-1)+XMIN
         PSI(J)=(1.0-X)**C+A1*(1.0-X)**(C+1)+A2*(1.0-X)**(C+2)
         J=J-1
15     CONTINUE
C
C      Calculate the last (PSIUPP-1) values of S.
C
       J=HIGH
       DO 20 I=POINTS,(POINTS-(PSIUPP-2)),-1
         S(I)=PSI(J-1)/PSI(J)
         J=J-1
20     CONTINUE
C
C      REASSIGN R(1) AND R(2) IF PSI IS ZERO AT THE LOWER BOUND.
C
       IF (BTYPE .EQ. 8) THEN
         R(1)=1.67D+200
         R(2)=U(2)/YMAT(2)
       ENDIF
C
C      REASSIGN S(POINTS-1) AND S(POINTS) IF PSI IS ZERO AT THE UPPER
C      BOUND.
C
```

```
      IF (BTYPE .EQ. 5) THEN
        S(POINTS-1)=U(POINTS-1)/ZMAT(POINTS-1)
        S(POINTS)=1.67D+200
      ENDIF
C
C     REASSIGN S(POINTS) AND S(POINTS-1) IF DERIVATIVE OF PSI AT
C     UPPER BOUND IS ZERO.
C
      IF (BTYPE .EQ. 6) THEN
        S(POINTS)=U(POINTS)/2.0
        S(POINTS-1)=U(POINTS-1)/ZMAT(POINTS-1)-(YMAT(POINTS-1)/
     X            ZMAT(POINTS-1))*(1.0/S(POINTS))
      ENDIF
C
C     REASSIGN R(1) AND R(2) IF DERIVATIVE OF PSI AT LOWER BOUND IS
C     ZERO.
C
      IF (BTYPE .EQ. 9) THEN
        R(1)=U(1)/2.0
        R(2)=U(2)/YMAT(2)-(ZMAT(2)/YMAT(2))*(1.0/R(1))
      ENDIF
    ENDIF
    RETURN
    END
C
C
C
    SUBROUTINE CYCLE1(E,EI,QN,LASTQN,K)
C
C   THIS SUBROUTINE ASSIGNS VALUES TO LASTQN AND EI.
C
C
    INTEGER*2 K,LASTQN,QN
    REAL*8 E(1),EI
C
C
    IF ((QN .NE. 0) .AND. (QN .NE. 1)) THEN
      EI=ABS(E(K)/QN)
    ELSE
      IF (QN .EQ. 1) THEN
        EI=ABS(0.15*E(K))
      ENDIF
    ENDIF
    LASTQN=QN
    RETURN
    END
C
C
C
    SUBROUTINE CYCLE3(E,EI,GUESS,QN,LASTQN,K,EPREV,LOOP,COUNT,POINTS,
   X            FUNCT,R,S,T,H,ALG,EVAL,WAY1,FOUND,GUESS2,MINVAL,
   X            A)
```

```
C
C
C    THIS SUBROUTINE ASSIGNS VALUES TO GUESS AND EI. IT ASSIGNS LOOP
C    TO BE TRUE IF THE NEXT EIGENVALUE HAS NOT BEEN FOUND. THE METHOD
C    (WAY1 OR NOT WAY1) FOR CALCULATING THE NEXT EIGENVALUE IS CHOSEN.
C    FOR SOME EIGENVALUES, THE METHOD MAY INITIALLY BE NEWTON'S METHOD
C    (WAY1), AND THEN BE CHANGED TO THE BISECTION METHOD (NOT WAY1)
C    WHEN CYCLE3 IS CALLED AGAIN LATER.
C
C
      INTEGER*2 ALG,COUNT,EVAL,K,LASTQN,POINTS,QN
      REAL*8 E(1),EI,GUESS,EPREV,AC,EHIGH,ELOW,H,T(1),S(1),R(1),
     X     FUNCT(1),E2,A1S,A1R,A2S,A2R,B2S,B2R,GUESS2,MINVAL,SPRIME,
     X     RPRIME,A(1)
      LOGICAL LOOP,WAY1,FOUND
C
C
      AC=3.0D-01
      IF ((COUNT .EQ. 0) .AND. (QN .NE. (LASTQN-1))) THEN
        IF (QN .GT. (LASTQN-1)) THEN
          WAY1=.TRUE.
          EI=1.4*EI
        ELSE
          IF (QN .EQ. (LASTQN-2)) THEN
            GUESS2=E(K)
            FOUND=.TRUE.
          ENDIF
          ELOW=E(K)
          EHIGH=EPREV
          GUESS=(EHIGH+ELOW)/2.0 + EI
          WAY1=.FALSE.
        ENDIF
        LOOP=.TRUE.
      ENDIF
      IF ((COUNT .GT. 0) .AND. (.NOT. WAY1)) THEN
        IF (QN .GT. (LASTQN-1)) THEN
          EHIGH=E(K)
        ELSE
          IF (QN .LT. (LASTQN-1)) THEN
            ELOW=E(K)
          ELSE
            LOOP=.TRUE.
            GUESS=E(K)+EI
            FUNCT(EVAL)=1.0
C
C
C           THE BLATT FORMULA (SEE [2] OF THE LONG WRITE-UP) IS
C           USED HERE. THE FORMULA IS:
C
C           E-Q=-psi(x0)*A(x0)*[dpsi/dx(x0,RIGHT)-dpsi/dx(x0,LEFT)]/K
C
C           WHERE K= integral_a^b psi*psi dx. Q IS THE CURRENT ESTIMATE FOR E.
```

```
C         ONLY THE SIGN OF E-Q IS NEEDED, AND THE EQUATION BECOMES
C
C         E-Q=-A(x0)*[dψ/dx(x0,RIGHT)-dψ/dx(x0,LEFT)]
C
C         SINCE THE INTEGRAL ,K, IS POSITIVE, AND ψ(x0) IS SET
C         EQUAL TO POSITIVE ONE.
C
          FUNCT(EVAL)=1.0
          IF (ALG .EQ. 1) THEN
            SPRIME=(0.5-T(EVAL+1))*(1.0/S(EVAL+1))-(0.5-T(EVAL-1))
     X           *S(EVAL)
            RPRIME=(0.5-T(EVAL+1))*R(EVAL)-(0.5-T(EVAL-1))*
     X           (1.0/R(EVAL-1))
          ELSE
            SPRIME=((0.5-T(EVAL+1))*(1.0-T(EVAL)))/((1.0-T(EVAL+1)
     X           )*S(EVAL+1))-((0.5-T(EVAL-1))*(1.0-T(EVAL))*
     X           S(EVAL))/(1.0-T(EVAL-1))
            RPRIME=(0.5-T(EVAL+1))*R(EVAL)*(1.0-T(EVAL))/
     X           (1.0-T(EVAL+1))-((0.5-T(EVAL-1))*(1.0-T(EVAL))
     X           )/(R(EVAL-1)*(1.0-T(EVAL-1)))
          ENDIF
          E2=-A(EVAL)*(SPRIME-RPRIME)
          IF (E2 .GE. 0.0) THEN
            ELOW=E(K)
          ELSE
            EHIGH=E(K)
          ENDIF
        ENDIF
      ENDIF
      IF (((EHIGH-ELOW) .LE. AC) .AND. (QN .EQ. (LASTQN-1))) THEN
        E(K)=(EHIGH+ELOW)/2.0
        GUESS=E(K) + EI
        WAY1=.TRUE.
        LOOP=.TRUE.
      ELSE
        GUESS=(EHIGH+ELOW)/2.0 + EI
        LOOP=.TRUE.
      ENDIF
    ELSE
      IF ((QN .GT. (LASTQN-1)) .AND. ((COUNT .NE. 0) .AND.
     X    (COUNT .NE. 1))) THEN
        EI=1.4*EI
        LOOP=.TRUE.
      ELSE
        IF (QN .EQ. (LASTQN-1)) THEN
          GUESS=E(K)
          EI=0.5*(EPREV-E(K))
          IF (QN .EQ. 1) THEN
            EI=0.2*(EPREV-E(K))
          ENDIF
        ELSE
```

```
            IF (COUNT .NE. 0) THEN
              IF (QN .EQ. (LASTQN-2)) THEN
                GUESS2=E(K)
                FOUND=.TRUE.
              ENDIF
              ELOW=E(K)
              EHIGH=EPREV
              GUESS=(EHIGH+ELOW)/2.0 + EI
              LOOP=.TRUE.
              WAY1=.FALSE.
            ENDIF
            IF ((COUNT .EQ. 1) .AND. (QN .GT. (LASTQN-1))) THEN
              WAY1=.FALSE.
              EHIGH=E(K)
              ELOW=MINVAL
              GUESS=(EHIGH+ELOW)/2.0+EI
              LOOP=.TRUE.
            ENDIF
          ENDIF
        ENDIF
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE CYCLE4(E,EI,GUESS,QN,LASTQN,K,EPREV,LOOP,COUNT,POINTS,
     X            FUNCT,R,S,T,H,ALG,EVAL,WAY1,FOUND,GUESS2,MINVAL,
     X            A)
C
C     THIS SUBROUTINE IS THE SAME AS CYCLE3, EXCEPT THAT IT IS SET UP
C     FOR CASES (BOUNDARY CONDITIONS 4. AND 5.) WHERE THE NODE COUNT
C     CHANGES BY TWO (BYTW01 OR BYTW02 = TRUE) BETWEEN EIGENVALUES.
C
C
      INTEGER*2 ALG,COUNT,EVAL,K,LASTQN,POINTS,QN
      REAL*8 E(1),EI,GUESS,EPREV,AC,EHIGH,ELOW,H,T(1),S(1),R(1),
     X     FUNCT(1),E2,A1S,A1R,A2S,A2R,B2S,B2R,GUESS2,MINVAL,A(1)
      LOGICAL LOOP,WAY1,FOUND
C
C
      AC=3.0D-01
      IF ((COUNT .EQ. 0) .AND. (QN .NE. (LASTQN-2))) THEN
        IF (QN .GT. (LASTQN-2)) THEN
          WAY1=.TRUE.
          EI=1.4*EI
        ELSE
          IF (QN .EQ. (LASTQN-4)) THEN
            GUESS2=E(K)
            FOUND=.TRUE.
          ENDIF
          ELOW=E(K)
          EHIGH=EPREV
```

```
          GUESS=(EHIGH+ELOW)/2.0 + EI
          WAY1=.FALSE.
        ENDIF
        LOOP=.TRUE.
      ENDIF
      IF ((COUNT .GT. 0) .AND. (.NOT. WAY1)) THEN
        IF (QN .GT. (LASTQN-2)) THEN
          EHIGH=E(K)
        ELSE
          IF (QN .LT. (LASTQN-2)) THEN
            ELOW=E(K)
          ELSE
            LOOP=.TRUE.
            GUESS=E(K)+EI
C
C           THE BLATT FORMULA (SEE [2] OF THE LONG WRITE-UP) IS
C           USED HERE. THE FORMULA IS:
C
C           E-Q=-psi(x0)*A(x0)*[dpsi/dx(x0,RIGHT)-dpsi/dx(x0,LEFT)]/K
C
C           WHERE K= integral_a^b psi*psi dx. Q IS THE CURRENT ESTIMATE FOR E.
C           ONLY THE SIGN OF E-Q IS NEEDED, AND THE EQUATION BECOMES
C
C           E-Q=-A(x0)*[dpsi/dx(x0,RIGHT)-dpsi/dx(x0,LEFT)]
C
C           SINCE THE INTEGRAL ,K, IS POSITIVE, AND psi(x0) IS SET
C           EQUAL TO POSITIVE ONE.
C
            FUNCT(EVAL)=1.0
            IF (ALG .EQ. 1) THEN
              SPRIME=(0.5-T(EVAL+1))*(1.0/S(EVAL+1))-(0.5-T(EVAL-1))
     X              *S(EVAL)
              RPRIME=(0.5-T(EVAL+1))*R(EVAL)-(0.5-T(EVAL-1))*
     X              (1.0/R(EVAL-1))
            ELSE
              SPRIME=((0.5-T(EVAL+1))*(1.0-T(EVAL)))/((1.0-T(EVAL+1)
     X              )*S(EVAL+1))-((0.5-T(EVAL-1))*(1.0-T(EVAL))*
     X              S(EVAL))/(1.0-T(EVAL-1))
              RPRIME=(0.5-T(EVAL+1))*R(EVAL)*(1.0-T(EVAL))/
     X              (1.0-T(EVAL+1))-((0.5-T(EVAL-1))*(1.0-T(EVAL))
     X              )/(R(EVAL-1)*(1.0-T(EVAL-1)))
            ENDIF
            E2=-A(EVAL)*(SPRIME-RPRIME)
            IF (E2 .GE. 0.0) THEN
              ELOW=E(K)
            ELSE
              EHIGH=E(K)
            ENDIF
          ENDIF
        ENDIF
```

```
          IF (((EHIGH-ELOW) .LE. AC) .AND. (QN .EQ. (LASTQN-2))) THEN
            E(K)=(EHIGH+ELOW)/2.0
            GUESS=E(K) + EI
            WAY1=.TRUE.
            LOOP=.TRUE.
          ELSE
            GUESS=(EHIGH+ELOW)/2.0 + EI
            LOOP=.TRUE.
          ENDIF
        ELSE
          IF ((QN .GT. (LASTQN-2)) .AND. ((COUNT .NE. 0) .AND.
     X      (COUNT .NE. 1))) THEN
            EI=1.4*EI
            LOOP=.TRUE.
          ELSE
            IF (QN .EQ. (LASTQN-2)) THEN
              GUESS=E(K)
              EI=0.5*(EPREV-E(K))
              IF (QN .EQ. 1) THEN
                EI=0.2*(EPREV-E(K))
              ENDIF
            ELSE
              IF (COUNT .NE. 0) THEN
                IF (QN .EQ. (LASTQN-4)) THEN
                  GUESS2=E(K)
                  FOUND=.TRUE.
                ENDIF
                ELOW=E(K)
                EHIGH=EPREV
                GUESS=(EHIGH+ELOW)/2.0 + EI
                LOOP=.TRUE.
                WAY1=.FALSE.
              ENDIF
              IF ((COUNT .EQ. 1) .AND. (QN .GT. (LASTQN-2))) THEN
                WAY1=.FALSE.
                EHIGH=E(K)
                ELOW=MINVAL
                GUESS=(EHIGH+ELOW)/2.0+EI
                LOOP=.TRUE.
              ENDIF
            ENDIF
          ENDIF
        ENDIF
        RETURN
        END
C
C
C
        SUBROUTINE EIVECS(FUNCT,R,S,T,H,ALG,POINTS,EVAL)
```

```
C
C
C    THIS SUBROUTINE CALCULATES THE VALUES OF A NORMALIZED EIGENFUNC-
C    TION AT EACH GRID POINT.
C
C
      INTEGER*2 ALG,EVAL,POINTS,I
      REAL*8 FUNCT(1),R(1),S(1),T(1),H,S2
C
C
      FUNCT(EVAL)=1.0
C
C    CALCULATE EIGENFUNCTION FOR B(x) NON-ZERO.
C
      IF (ALG .EQ. 1) THEN
        DO 5 I=EVAL+1,POINTS
          FUNCT(I)=FUNCT(I-1)/S(I)
5       CONTINUE
        DO 10 I=EVAL,1,-1
          FUNCT(I)=FUNCT(I+1)/R(I)
10      CONTINUE
      ENDIF
C
C    CALCULATE EIGENFUNCTION FOR B(x) =0.
C
      IF (ALG .EQ. 2) THEN
        DO 35 I=EVAL+1,POINTS
          FUNCT(I)=FUNCT(I-1)/S(I)
35      CONTINUE
        DO 40 I=EVAL,1,-1
          FUNCT(I)=FUNCT(I+1)/R(I)
40      CONTINUE
        DO 42 I=1,POINTS
          FUNCT(I)=FUNCT(I)/(1.0-T(I))
42      CONTINUE
      ENDIF
      DO 45 I=1,POINTS
        T(I)=FUNCT(I)**2
45    CONTINUE
      CALL SIMPLE(T,POINTS,H,S2)
      S2=1.0/SQRT(S2)
C
C    NORMALIZE EIGENFUNCTION.
C
      DO 50 I=1,POINTS
        FUNCT(I)=S2*FUNCT(I)
50    CONTINUE
      RETURN
      END
```

```
C
C
C
      SUBROUTINE SIMPLE(G,POINTS,H,S)
C
C
C     THIS SUBROUTINE CALCULATES THE INTEGRAL OF THE FUNCTION G FROM
C     XMIN TO XMAX, AND STORES IT IN S.
C
C
      INTEGER*2 POINTS,L
      REAL*8 G(1),H,S
C
C
      S=G(1)+4.0*G(2)+G(POINTS)
      L=1
10     L=L+2
        IF (L .GT. (POINTS-1)) THEN
          S=S*H/3.0
          RETURN
        ELSE
          S=S+2*G(L)+4*G(L+1)
          GOTO 10
        ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE STAT3(FIRST2,FUNCT,POINTS,DISK,E,K,FNAME,QN)
C
C
C     THIS SUBROUTINE SAVES THE EIGENFUNCTIONS AND EIGENVALUES OF A
C     GIVEN D.E.. THE EIGENFUNCTIONS AND CORRESPONDING
C     EIGENVALUES AND QUANTUM NUMBERS ARE STORED CONSECUTIVELY IN A
C     FILE CHOSEN PREVIOUSLY (IN SETUP) BY THE USER.
C
C
      INTEGER*2 POINTS,DISK,K,QN,I
      REAL*8 FUNCT(1),E(1),QNREAL
      CHARACTER FNAME*20
      LOGICAL FIRST2
C
C
      DISK=5
      QNREAL=QN
      WRITE(DISK,12) QNREAL,E(K)
12     FORMAT(F3.0,8X,F22.4,' ')
      WRITE(DISK,15) (FUNCT(I),I=1,POINTS)
15     FORMAT(4(F8.5,' '))
      FIRST2=.FALSE.
      RETURN
      END
```

```
C
C
C
      SUBROUTINE STAT4(FUNCT,POINTS,DISK,FNAME,XMIN,H)
C
C
C     THIS SUBROUTINE SAVES TO DISK THE X VALUES (AT EACH GRID POINT)
C     OF A GIVEN D.E.. 'DUMMY' IS WRITTEN TO DISK AS WELL TO MAKE THIS
C     OUTPUT OF THE SAME FORM AS THE EIGENFUNCTION OUTPUT.
C
C
      INTEGER*2 POINTS,DISK,I
      REAL*8 FUNCT(1),XMIN,H,RPOINT,DUMMY
      CHARACTER FNAME*20
C
C
      DISK=5
      OPEN(DISK,FILE=FNAME,STATUS='NEW')
      RPOINT=POINTS
      WRITE(DISK,5) RPOINT,DUMMY
5     FORMAT(F6.0,' ',F4.0,' ')
      DO 10 I=1,POINTS
        FUNCT(I)=XMIN+(I-1)*H
10    CONTINUE
      WRITE(DISK,15) (FUNCT(I),I=1,POINTS)
15    FORMAT(4(F8.5,' '))
      RETURN
      END
C
C
C
      SUBROUTINE EVIEW(EIVAL,ESIZE,ENAME,ECOUNT,ERROR,POINTS,XMIN,XMAX,
     X                 ANAME,BNAME,VNAME,EMIN,EMAX,LBC,UBC)
C
C
C     THIS SUBROUTINE PRINTS THE CALCULATED EIGENVALUES AND NODE COUNTS
C     ON THE SCREEN OR THE PRINTER. IT ALSO ECHOES THE INPUT DATA.
C
C
      INTEGER*2 ESIZE,ECOUNT,FILENU,I,CHOICE,LBC,UBC,POINTS,PRINTE,VIDEO
      REAL*8 EIVAL(ESIZE,1),EMIN,EMAX,XMIN,XMAX
      CHARACTER ENAME*20,ANAME*18,BNAME*18,VNAME*20
      LOGICAL ERROR
C
C
      VIDEO=0
      PRINTE=4
5     IF (.NOT. ERROR) THEN
        CALL WRITEL(25)
        WRITE(*,*) '        WHERE DO YOU WANT THE OUTPUT SENT ?'
        WRITE(*,*)
        WRITE(*,*) '        1.) THE PRINTER'
        WRITE(*,*) '        2.) THE SCREEN'
```

```
      CALL WRITEL(10)
      READ(*,*) CHOICE
      IF ((CHOICE .NE. 1) .AND. (CHOICE .NE. 2)) THEN
        CALL WRITEL(25)
        WRITE(*,*) '*** ERROR ***'
        WRITE(*,*)
        WRITE(*,*) 'YOU CAN ONLY CHOOSE THE PRINTER OR THE SCREEN.'
        CALL WRITEL(10)
        CALL HOLD
        CALL WRITEL(25)
        GOTO 5
      ENDIF
      IF (CHOICE .EQ. 1) THEN
        OPEN(PRINTER,FILE='LPT1:')
        FILENU=PRINTE
      ELSE
        FILENU=VIDEO
      ENDIF
      DO 10 I=1,5
        WRITE(FILENU,*)
10     CONTINUE
      CALL WRITEL(25)
      WRITE(FILENU,*) '         **********************************
     X******************'
      WRITE(FILENU,15) ENAME
15     FORMAT('          ***    THE ',A20,' EQUATION    ***')
      WRITE(FILENU,*) '         **********************************
     X******************'
      WRITE(FILENU,*)
      WRITE(FILENU,17) ANAME,BNAME
17     FORMAT('          A(x) = ',A18,' B(x) = ',A18)
      WRITE(FILENU,*)
      WRITE(FILENU,18) VNAME
18     FORMAT('                   V(x) = ',A20)
      WRITE(FILENU,*)
      WRITE(FILENU,20) POINTS,XMIN,XMAX
20     FORMAT('          ',I4,' POINTS    XMIN = ',F8.4,'   XMAX
     X = ',F8.4)
      WRITE(FILENU,*)
      IF ((LBC .EQ. 1) .AND. (UBC .EQ. 1)) THEN
        WRITE(FILENU,21) XMIN,XMAX
21       FORMAT('           PSI(',F10.3,') = 0.0',3X,'PSI(',F10
     X.3,') = 0.0')
      ENDIF
      IF ((LBC .EQ. 1) .AND. (UBC .EQ. 2)) THEN
        WRITE(FILENU,22) XMIN,XMAX
22       FORMAT('          PSI(',F10.3,') = 0.0',3X,'dPSI/dX(',F
     X10.3,') = 0.0')
      ENDIF
```

```
      IF ((LBC .EQ. 1) .AND. (UBC .EQ. 3)) THEN
        WRITE(FILENU,23) XMIN,XMAX
23       FORMAT('          PSI(',F7.3,') = 0.0',1X,'FROB. EXP. OF
     X PSI AT X = ',F7.3)
      ENDIF
      IF ((LBC .EQ. 2) .AND. (UBC .EQ. 1)) THEN
        WRITE(FILENU,24) XMIN,XMAX
24       FORMAT('          dPSI/dX(',F10.3,') = 0.0',3X,'PSI(',F
     X10.3,') = 0.0')
      ENDIF
      IF ((LBC .EQ. 2) .AND. (UBC .EQ. 2)) THEN
        WRITE(FILENU,25) XMIN,XMAX
25       FORMAT('          dPSI/dX(',F9.3,') = 0.0',3X,'dPSI/dX('
     X,F9.3,') = 0.0')
      ENDIF
      IF ((LBC .EQ. 2) .AND. (UBC .EQ. 3)) THEN
        WRITE(FILENU,26) XMIN,XMAX
26       FORMAT('          dPSI/dX(',F7.3,') =0.0',1X,'FRO. EX. O
     XF PSI AT X =',F7.3)
      ENDIF
      IF ((LBC .EQ. 3) .AND. (UBC .EQ. 1)) THEN
        WRITE(FILENU,27) XMIN,XMAX
27       FORMAT('          FROB. EXP. OF PSI AT X = ',F7.3,1X,'PS
     XI(',F7.3,') = 0.0')
      ENDIF
      IF ((LBC .EQ. 3) .AND. (UBC .EQ. 2)) THEN
        WRITE(FILENU,28) XMIN,XMAX
28       FORMAT('          FRO. EXP. OF PSI AT X=',F7.3,1X,'dPSI/
     XdX(',F7.3,') =0.0')
      ENDIF
      IF ((LBC .EQ. 3) .AND. (UBC .EQ. 3)) THEN
        WRITE(FILENU,29) XMIN,XMAX
29       FORMAT('          FRO. EXP. AT X=',F7.3,1X,'FRO. EX. OF
     XPSI AT X=',F7.3)
      ENDIF
      IF (LBC .EQ. 4) THEN
        WRITE(FILENU,30) XMIN,XMAX
30       FORMAT('            PSI(',F10.3,') = PSI(',F10.3,')
     X AND')
        WRITE(FILENU,31) XMIN,XMAX
31       FORMAT('          dPSI/dX(',F10.3,') = dPSI/dX(',F10.3
     X,') = 0.0')
      ENDIF
      IF (LBC .EQ. 5) THEN
        WRITE(FILENU,32) XMIN,XMAX
32       FORMAT('          PSI(',F10.3,') = PSI(',F10.3,') = 0
     X.0  AND')
        WRITE(FILENU,33) XMIN,XMAX
33       FORMAT('            dPSI/dX(',F10.3,') = dPSI/dX(',F1
     X0.3,')')
      ENDIF
      WRITE(FILENU,*)
      WRITE(FILENU,35) EMIN,EMAX
```

```
35     FORMAT('           EIGENVALUE LIMITS: ',F12.4,',',3X,F12.4
     X)
       WRITE(FILENU,*)
       WRITE(FILENU,*) '              NODES OF'
       WRITE(FILENU,*) '            EIGENFUNCTIONS
     XEIGENVALUES'
       WRITE(FILENU,*) '            ==============
     X============'
       WRITE(FILENU,40) EIVAL(1,1),EIVAL(1,2)
       IF ((FILENU .EQ. 0) .AND. (ECOUNT .GE. 2)) THEN
         CALL WRITEL(2)
         PAUSE 'HIT RETURN TO SEE THE REMAINING EIGENVALUE(S).'
         CALL WRITEL(2)
       ENDIF
       DO 55 I=2,ECOUNT
         IF (FILENU .EQ. 0) THEN
           IF (MOD(I,10) .NE. 0) THEN
             WRITE(FILENU,40) EIVAL(I,1),EIVAL(I,2)
40            FORMAT('0          ',F12.1,'          ',F19.4)
           ELSE
             CALL WRITEL(2)
             PAUSE 'HIT RETURN TO SEE THE REMAINING EIGENVALUE(S).'
             CALL WRITEL(2)
             WRITE(FILENU,45) EIVAL(I,1),EIVAL(I,2)
45            FORMAT('0          ',F12.1,'          ',F19.4)
           ENDIF
         ELSE
           WRITE(FILENU,50) EIVAL(I,1),EIVAL(I,2)
50          FORMAT('0          ',F12.1,'          ',F19.4)
         ENDIF
55     CONTINUE
       IF (CHOICE .EQ. 1) THEN
         CLOSE(PRINTER)
       ENDIF
     ELSE
       CALL WRITEL(12)
       WRITE(*,*) 'THE EIGENVALUE(S) DIDN''T FALL WITHIN THE GIVEN LIM
     XITS !!!'
       CALL WRITEL(10)
     ENDIF
     RETURN
     END
```

Appendix D

Test and Utility Subroutines

A test program, *TSTS*, was written in order that the various transformations employed in *NPLOT* could be checked. The program is set up for three and four atom molecules only. *TSTS* requires an O matrix (see chapter 2) to be recalled from disk, so one must have been previously stored on disk. The user indicates the number of coordinates for the molecule, chooses the number of grid points at which the testing is done, and chooses the two variable coordinates. As well, equilibrium values for the other coordinates are chosen.

This program first carries out the transformation from the target coordinates (e.g. BRI or Hypersphericals) to bond distance coordinates. The inverse transformation is then done, and the new values for the target coordinates are compared to the original values. If the coordinate values do not agree to within a small constant value, an error message is printed.

The transformation to bond distance coordinates is then done, and these values are compared to the first calculated set of bond distance coordinates. Again, if there is any discrepancy, an error message is printed.

The entire procedure outlined above is performed at each grid point for the two variable coordinates.

To facilitate running *TSTS*, a batch file was written, which copies the object code of the particular transformation (e.g. BRI ,polar etc.) from floppy disk and links it with the rest of the program. When linking is finished, the program can be run.

Mainly for the purposes of testing the program *GJVGEN*, three small test programs were used.

The first of these is the program *MULT*, which prompts the user for the elements of two n x n matrices, multiplies the two matrices together, and prints their product.

The second program, *INVER*, prompts the user for the elements of an n x n matrix, and then prints the matrice's inverse.

The third program, *PSQRT*, prompts the user for the elements of an n x n matrix, and prints the matrice's positive square root.

A batch file was written which copies from floppy disk the object code of one of the above three programs and links it with the FORTRAN libraries. An executable file is then ready to be run.

```
$DEBUG
$NOFLOATCALLS
$STORAGE:2
C
C
C
      PROGRAM TSTS
C
C
      INTEGER*2 FIR,SEC,I,J,K,MAX4,POINTS,NCOORD,N,ROWS,COLS
      REAL*8 AC,S(4,4),XMIN(9),XMAX(9),H(9),Q1(9),Q2(9),X1(9),X2(9)
      LOGICAL ERROR
      CHARACTER ROWNME(4)*15,COLNME(4)*5,MOLNAM*20,TRNAME*20,SNAME*20,
     X       LORB*1,CORG*1
C
C
      AC=1.0D-06
      MAX4=4
      WRITE(*,*) 'MOLECULE IS LINEAR OR BRANCHED ? (L/B)'
      READ(*,2) LORB
2     FORMAT(A1)
      WRITE(*,*) 'GIVE THE NUMBER OF COORDINATES.'
      READ(*,*) NCOORD
      N=(NCOORD+6)/3
      IF (N .EQ. 4) THEN
        WRITE(*,*) 'CHA FRAME OR 3 GJV FRAME ? (C/G)'
        READ(*,4) CORG
4       FORMAT(A1)
      ENDIF
      ROWS=N-1
      COLS=N-1
      WRITE(*,*) 'RECALL AN O MATRIX FROM DISK.'
      CALL RECALL(S,ROWNME,COLNME,MOLNAM,TRNAME,SNAME,MAX4,ROWS,COLS)
      WRITE(*,*) 'GIVE THE TARGET COORDINATES WHICH ARE VARIABLE.'
      READ(*,*) FIR,SEC
      WRITE(*,*) 'GIVE THE NUMBER OF GRID POINTS.'
      READ(*,*) POINTS
5     WRITE(*,10) FIR
10    FORMAT(' GIVE THE RANGE FOR TARGET COORDINATE ',I1,'.')
      READ(*,*) XMIN(FIR),XMAX(FIR)
      ERROR=.FALSE.
      H(FIR)=(XMAX(FIR)-XMIN(FIR))/(POINTS-1)
      WRITE(*,15) SEC
15    FORMAT(' GIVE THE RANGE FOR TARGET COORDINATE ',I1,'.')
      READ(*,*) XMIN(SEC),XMAX(SEC)
      H(SEC)=(XMAX(SEC)-XMIN(SEC))/(POINTS-1)
      DO 25 I=1,NCOORD
        IF ((I .NE. FIR) .AND. (I .NE. SEC)) THEN
          WRITE(*,20) I
20          FORMAT(' GIVE XEQ(',I1,') .')
          READ(*,*) X1(I)
        ENDIF
25    CONTINUE
```

```
C
C     SCAN THROUGH RANGE OF FIRST VARIABLE COORDINATE.
C
      DO 45 I=1,POINTS
        X1(FIR)=XMIN(FIR)+(I-1)*H(FIR)
C
C       SCAN THROUGH RANGE OF SECOND VARIABLE COORDINATE.
C
        DO 40 J=1,POINTS
          X1(SEC)=XMIN(SEC)+(J-1)*H(SEC)
          CALL XTOQ(X1,Q1,N,S,MAX4,LORB,ERROR,CORG)
          IF (ERROR) THEN
            GOTO 5
          ENDIF
          CALL QTOX(Q1,X2,N,S,MAX4,LORB,CORG)
          IF (ERROR) THEN
            GOTO 5
          ENDIF
C
C         CHECK THAT TRANSFORMATION HAS BEEN COORECT.
C
          DO 30 K=1,NCOORD
            IF (ABS(X2(K)-X1(K)) .GT. AC) THEN
              IF (((K .LE. 2) .AND. (N .EQ. 3)) .OR. ((K .LE. 3)
     X            .AND. (N .EQ. 4))) THEN
                WRITE(*,*) 'X2(',K,') IS:',X2(K)
                WRITE(*,*) 'X1(',K,') IS:',X1(K)
              ELSE
                IF (ABS(COS(X2(K))-COS(X1(K))) .GT. AC) THEN
                  WRITE(*,*) 'COS(X2(',K,')) IS:',COS(X2(K))
                  WRITE(*,*) 'COS(X1(',K,')) IS:',COS(X1(K))
                ENDIF
              ENDIF
            ENDIF
30        CONTINUE
          CALL XTOQ(X2,Q2,N,S,MAX4,LORB,ERROR,CORG)
C
C         CHECK THAT INVERSE TRANSFORMATION HAS BEEN CORRECT.
C
          DO 35 K=1,NCOORD
            IF (ABS(Q2(K)-Q1(K)) .GT. AC) THEN
              WRITE(*,*) 'Q2(',K,') IS:',Q2(K)
              WRITE(*,*) 'Q1(',K,') IS:',Q1(K)
            ENDIF
35        CONTINUE
40      CONTINUE
45    CONTINUE
      STOP
      END
```

```
C
C
C
      SUBROUTINE QTOX(Q,X,N,S,MAX4,LORB,CORG)
C
C
      INTEGER*2 N,MAX4,A
      REAL*8 Y(4,4),Q(1),S2(9),X(1),S(MAX4,1),TEMP1(4,4),TEMP2(4,4),
     X     TEMP3(4,4),TEMP4(4,4),EXTRA(4,4)
      CHARACTER TRNAME*20,LORB*1,CORG*1
C
C
C     CALCULATE TARGET COORDINATES FROM BOND DISTANCE-BOND ANGLE
C     COORDINATES.
C
      MAX4=4
      CALL FSOURC(Q,S2,N,LORB)
      CALL FORM(S2,Y,N,MAX4)
      A=1
      CALL TRFORM(Y,N,A,S,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,EXTRA,TRNAME)
      CALL FORMIN(Y,X,N,MAX4)
C
C     IF CHA FRAME IS EMPLOYED, CALCULATE ANGLE PHI.
C
      IF ((N .EQ. 4) .AND. (CORG .EQ. 'C')) THEN
        CALL TOPHI(X)
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE XTOQ(X,Q,N,S,MAX4,LORB,ERROR,CORG)
C
C
      INTEGER*2 N,MAX4,A,NCOORD
      REAL*8 X2(9),Y(4,4),Q(1),S2(9),X(1),S(MAX4,1),TEMP1(4,4),
     X     TEMP2(4,4),TEMP3(4,4),TEMP4(4,4),EXTRA(4,4)
      LOGICAL ERROR
      CHARACTER TRNAME*20,LORB*1,CORG*1
C
C
C     CALCULATE BOND DISTANCE-BOND ANGLE COORDINATES FROM TARGET
C     COORDINATES.
C
      NCOORD=3*N-6
      MAX4=4
      CALL ASSIGV(Q,X,NCOORD)
C
C     CALCULATE THIRD ANGLE FROM ANGLE PHI, IF CHA FRAME IS EMPLOYED.
C
```

```
      IF ((N .EQ. 4) .AND. (CORG .EQ. 'C')) THEN
        CALL FROMPH(Q,ERROR)
      ENDIF
      CALL FORM(Q,Y,N,MAX4)
      A=2
      CALL TRFORM(Y,N,A,S,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,EXTRA,TRNAME)
      CALL FORMIN(Y,X2,N,MAX4)
      CALL TSOURC(X2,Q,N,LORB,ERROR)
      RETURN
      END
C
C
C
      SUBROUTINE FORM(X,Y,N,MAX4)
C
C
      INTEGER*2 N,MAX4,I,J,COUNT
      REAL*8 X(1),Y(MAX4,1)
C
C
C     FORM MATRIX OF TARGET COORDINATE VALUES.
C
      COUNT=0
      DO 10 I=1,(N-1)
        COUNT=COUNT+1
        Y(COUNT,COUNT)=X(COUNT)
10    CONTINUE
      DO 30 I=1,(N-1)
        DO 20 J=I+1,(N-1)
          COUNT=COUNT+1
          Y(I,J)=X(COUNT)
20      CONTINUE
30    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE FSOURC(Q,QSANG,N,LORB)
C
C
      INTEGER*2 N,I
      REAL*8 Q(1),QSANG(1),COSANG,PI
      CHARACTER LORB*1
C
C
C     CALCULATE BOND DISTANCE-BOND ANGLE COORDINATES FROM BOND DISTANCE
C     COORDINATES.
C
     PI=ACOS(-1.0)
      DO 5 I=1,(N-1)
        QSANG(I)=Q(I)
5     CONTINUE
```

```
      IF (N .EQ. 3) THEN
        COSANG=(Q(2)**2+Q(1)**2-Q(3)**2)/(2.0*Q(2)*Q(1))
        CALL ACHECK(COSANG)
        QSANG(3)=ACOS(COSANG)
      ENDIF
      IF ((N.EQ. 4) .AND. (LORB .EQ. 'B')) THEN
        COSANG=(Q(1)**2+Q(2)**2-Q(4)**2)/(2.0*Q(2)*Q(1))
        CALL ACHECK(COSANG)
        QSANG(4)=ACOS(COSANG)
        COSANG=(Q(1)**2+Q(3)**2-Q(5)**2)/(2.0*Q(1)*Q(3))
        CALL ACHECK(COSANG)
        QSANG(5)=ACOS(COSANG)
        COSANG=(Q(2)**2+Q(3)**2-Q(6)**2)/(2.0*Q(2)*Q(3))
        CALL ACHECK(COSANG)
        QSANG(6)=ACOS(COSANG)
      ENDIF
      IF ((N .EQ. 4) .AND. (LORB .EQ. 'L')) THEN
        COSANG=(Q(1)**2+Q(2)**2-Q(4)**2)/(2.0*Q(2)*Q(1))
        CALL ACHECK(COSANG)
        QSANG(4)=ACOS(COSANG)
        COSANG=(-Q(5)**2+Q(1)**2+Q(3)**2)/(2.0*Q(1)*Q(3))
        CALL ACHECK(COSANG)
        QSANG(5)=PI-ACOS(COSANG)
        COSANG=(Q(4)**2+Q(5)**2-Q(6)**2-Q(1)**2)/(2.0*Q(2)*Q(3))
        CALL ACHECK(COSANG)
        QSANG(6)=ACOS(COSANG)
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE TRFORM(Y,N,A,S,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,EXTRA,
     X                  TRNAME)
C
C
      INTEGER*2 N,A,MAX4,I,J
      REAL*8 S(MAX4,1),Y(MAX4,1),TEMP1(MAX4,1),
     X     TEMP2(MAX4,1),TEMP3(MAX4,1),TEMP4(MAX4,1),EXTRA(MAX4,1)
      CHARACTER TRNAME*20
C
C
C     FORM G OR G' AND CALCULATE OGO^t OR O^-1 G'(O^-1)^t.
C
      IF (A .EQ. 2) THEN
        CALL TOJAC(Y,N,MAX4,TEMP1,TEMP2)
      ENDIF
      CALL SINVER(Y,TEMP1,TEMP2,MAX4,N)
      CALL TRANSF(Y,S,N,A,MAX4,TEMP1,TEMP2,TEMP3)
      CALL TDEGRA(Y,N,MAX4,TEMP1)
```

```
      IF (A .EQ. 1) THEN
        CALL FROMJA(Y,N,MAX4,TEMP1,TEMP2,TEMP3,TEMP4,EXTRA,TRNAME)
      ENDIF
      RETURN
      END
C
C
C
      SUBROUTINE SINVER(Y,P,TEMP,MAX4,N)
C
C
      INTEGER*2 N,I,J,K,MAX4,D
      REAL*8 TEMP(MAX4,1),P(MAX4,1),Y(MAX4,1),SCALAR
C
C
C     FORM GRAM MATRIX (G OR G').
C
      D=N-1
      SCALAR=0.0
      CALL MATSCA(TEMP,SCALAR,P,D,MAX4)
      CALL ASSIGN(P,TEMP,D,MAX4)
      DO 120 I=1,D
        J=I
100     IF (I .EQ. J) THEN
          P(I,J)=Y(I,J)**2.0
        ENDIF
        IF (I .NE. J) THEN
          P(I,J)=COS(Y(I,J))*(Y(I,I))*Y(J,J)
        ENDIF
        P(J,I)=P(I,J)
        J=J+1
       IF (J .LE. D) THEN
         GOTO 100
       ENDIF
120    CONTINUE
      CALL ASSIGN(Y,P,D,MAX4)
      RETURN
      END
C
C
C
      SUBROUTINE TRANSF(M,S,N,A,MAX4,SP,ST,P)
C
C
      INTEGER*2 MAX4,D,A,N
      REAL*8 M(MAX4,1),S(MAX4,1),SP(MAX4,1),DET,ST(MAX4,1),
     X     P(MAX4,1)
C
```

```
C
C     CALCULATE OGO^t OR O^-1 G' (O^-1)^t.
C
      D=N-1
      CALL ASSIGN(SP,S,D,MAX4)
      IF (A .NE. 1) THEN
        CALL INVDET(P,SP,D,MAX4,DET,ST)
        CALL ASSIGN(SP,P,D,MAX4)
      ENDIF
      CALL MATRAN(ST,SP,D,MAX4)
      CALL MATMUL(P,M,ST,D,MAX4)
      CALL MATMUL(M,SP,P,D,MAX4)
      RETURN
      END
C
C
C
      SUBROUTINE TDEGRA(M,N,MAX4,A)
C
C
      INTEGER*2 MAX4,N,I,J,D
      REAL*8 M(MAX4,1),A(MAX4,1)
C
C
C     CALCULATE BOND DISTANCE-BOND ANGLE COORDINATES FROM GRAM MATRIX,
C     OR CALCULATE BASIC ROTATIONAL INVARIANT COORDINATES FROM GRAM
C     MATRIX.
C
      D=N-1
      DO 20 I=1,N-1
        J=I
10      IF (I .EQ. J) THEN
          A(I,J)=M(I,J)**(1.0/2.0)
        ENDIF
        IF (I .NE. J) THEN
          A(I,J)=M(I,J)/(M(I,I)*M(J,J))**(1.0/2.0)
        ENDIF
        IF ((I .NE. J) .AND. (A(I,J) .GT. 1.0)) THEN
          A(I,J)=1.0
        ENDIF
        IF ((I .NE. J) .AND. (A(I,J) .LT. -1.0)) THEN
          A(I,J) = -1.0
        ENDIF
        IF (I .NE. J) THEN
          A(I,J)=ACOS(A(I,J))
        ENDIF
        A(J,I)=A(I,J)
        IF (J .LT. (N-1)) THEN
          J=J+1
          GOTO 10
        ENDIF
20    CONTINUE
```

```
      CALL ASSIGN(M,A,D,MAX4)
      RETURN
      END
C
C
C
      SUBROUTINE FORMIN(Y,X,N,MAX4)
C
C
      INTEGER*2 I,J,N,MAX4,COUNT
      REAL*8 X(1),Y(MAX4,1)
C
C     ASSIGN THE INDIVIDUAL COORDINATES TO X, FROM MATRIX FORM IN Y.
C
      COUNT=0
      DO 10 I=1,(N-1)
        COUNT=COUNT+1
        X(COUNT)=Y(COUNT,COUNT)
10    CONTINUE
      DO 30 I=1,(N-1)
        DO 20 J=I+1,(N-1)
          COUNT=COUNT+1
          X(COUNT)=Y(I,J)
20      CONTINUE
30    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE TSOURC(QSANG,Q,N,LORB,ERROR)
C
C
      INTEGER*2 N
      REAL*8 QSANG(1),Q(1),PI
      LOGICAL ERROR
      CHARACTER LORB*1
C
C
C     CALCULATE BOND DISTANCE COORDINATES FROM BOND DISTANCE-BOND ANGLE
C     COORDINATES.
C
      PI=ACOS(-1.0)
      DO 5 I=1,(N-1)
        Q(I)=QSANG(I)
5     CONTINUE
      IF (N .EQ. 3) THEN
        Q(3)=(QSANG(1)**2.0+QSANG(2)**2.0-2.0*QSANG(1)*QSANG(2)*
     X       COS(QSANG(3)))**(1.0/2.0)
      ENDIF
```

```
      IF (N .EQ. 4) THEN
        IF (LORB .EQ. 'L') THEN
          IF ((QSANG(4) .EQ. PI) .AND. (QSANG(6) .NE.
     X        (PI - QSANG(5)))) THEN
            ERROR=.TRUE.
            WRITE(*,*) 'ERROR !! INCORRECT CONFIGURATION OF MOLECULE.
     X'
            GOTO 10
          ENDIF
          IF ((QSANG(5) .EQ. 0.0) .AND. (QSANG(4) .NE. QSANG(6))) THEN
            ERROR=.TRUE.
            WRITE(*,*) 'ERROR !! INCORRECT CONFIGURATION OF MOLECULE.
     X'
            GOTO 10
          ENDIF
        ENDIF
C
C       CHECK THAT ANGLES OBEY 'GEOMETRY RULES.'
C
        IF ((QSANG(4)+QSANG(6)+QSANG(5)) .GT. (2.0*PI)) THEN
          ERROR=.TRUE.
          WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
          GOTO 10
        ENDIF
        IF ((QSANG(4)+QSANG(6)) .LT. QSANG(5)) THEN
          ERROR=.TRUE.
          WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
          GOTO 10
        ENDIF
        IF ((QSANG(4)+QSANG(5)) .LT. QSANG(6)) THEN
          ERROR=.TRUE.
          WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
          GOTO 10
        ENDIF
        IF ((QSANG(6)+QSANG(5)) .LT. QSANG(4)) THEN
          ERROR=.TRUE.
          WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
          GOTO 10
        ENDIF
        IF ((N .EQ. 4) .AND. (LORB .EQ. 'B')) THEN
          Q(4)=(QSANG(1)**2.0+QSANG(2)**2.0-2.0*QSANG(1)*QSANG(2)*
     X        COS(QSANG(4)))**(1.0/2.0)
          Q(5)=(QSANG(1)**2.0+QSANG(3)**2.0-2.0*QSANG(1)*QSANG(3)*
     X        COS(QSANG(5)))**(1.0/2.0)
          Q(6)=(QSANG(2)**2.0+QSANG(3)**2.0-2.0*QSANG(2)*QSANG(3)*
     X        COS(QSANG(6)))**(1.0/2.0)
        ENDIF
        IF ((N .EQ. 4) .AND. (LORB .EQ. 'L')) THEN
          Q(4)=(QSANG(1)**2.0+QSANG(2)**2.0-2.0*QSANG(1)*QSANG(2)*
     X        COS(QSANG(4)))**(1.0/2.0)
          Q(5)=(Q(1)**2.0+Q(3)**2.0-2.0*QSANG(1)*QSANG(3)*
     X        COS(PI-QSANG(5)))**(1.0/2.0)
         Q(6)=(Q(4)**2.0+Q(5)**2.0-QSANG(1)**2.0-
```

```
     X        2.0*QSANG(2)*QSANG(3)*COS(QSANG(6)))**(1.0/2.0)
       ENDIF
     ENDIF
10    RETURN
     END
C
C
C
     SUBROUTINE ACHECK(COSANG)
C
C
     REAL*8 COSANG
C
C
C    AVOID ROUND OFF ERROR IN CALCULATION OF COSINE OF ANGLES.
C
     IF (COSANG .LT. -1.0) THEN
       COSANG=-1.0
     ENDIF
     IF (COSANG .GT. 1.0) THEN
       COSANG=1.0
     ENDIF
     RETURN
     END
C
C
C
     SUBROUTINE TOPHI(X)
C
C
     REAL*8 X(1),DENOM,NUMER,PI,COSANG,AC
C
C    CALCULATE ANGLE PHI FOR CHA FRAME.
C
     PI=ACOS(-1.0)
     AC=1.0D-09
     IF ((SIN(X(4)) .LT. AC) .OR. (SIN(X(5)) .LT. AC)) THEN
       X(6)=PI
     ELSE
       COSANG=(COS(X(6))-COS(X(4))*COS(X(5)))/(SIN(X(4))*SIN(X(5)))
       CALL ACHECK(COSANG)
       X(6)=ACOS(COSANG)
     ENDIF
     RETURN
     END
C
C
C
     SUBROUTINE FROMPH(X,ERROR)
C
C
     REAL*8 X(1),AC2,PI
     LOGICAL ERROR
```

```
C
C
C    TRANSFORM FROM ANGLE PHI TO ANGLE Θ23.
C
     AC2=1.0D-08
     AC=1.0D-05
     PI=ACOS(-1.0)
     IF ((ABS(SIN(X(4))) .LT. AC2) .OR. (ABS(SIN(X(5))) .LT.
    X  AC2)) THEN
       X(6)=ACOS(COS(X(4))*COS(X(5)))
     ELSE
       X(6)=ACOS(SIN(X(4))*SIN(X(5))*COS(X(6))+COS(X(4))*COS(X(5)))
     ENDIF
C
C    CHECK THAT ANGLES OBEY 'GEOMETRY RULES.'
C
     IF ((X(4)+X(6)+X(5)) .GT. (2.0*PI)) THEN
       ERROR=.TRUE.
       WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
       GOTO 10
     ENDIF
     IF ((X(4)+X(6)) .LT. X(5)) THEN
       ERROR=.TRUE.
       WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
       GOTO 10
     ENDIF
     IF ((X(4)+X(5)) .LT. X(6)) THEN
       ERROR=.TRUE.
       WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
       GOTO 10
     ENDIF
     IF ((X(6)+X(5)) .LT. X(4)) THEN
       ERROR=.TRUE.
       WRITE(*,*) 'INCORRECT RANGE OF ANGLE(S) !!!'
       GOTO 10
     ENDIF
10    RETURN
     END
C
C
C
$DEBUG
$NOFLOATCALLS
$STORAGE:2
C
C
C
     PROGRAM MULT
C
C    This program calculates the product of two n x n matrices.
C
     INTEGER*2 I,J,K,SIZE,RANGE
```

```
      REAL*8 M1(30,30),M2(30,30),NEW(30,30)
C
C
      WRITE(*,*) 'GIVE N.'
      READ(*,*) RANGE
      DO 80 I=1,RANGE
        DO 70 J=1,RANGE
          WRITE(*,*) 'GIVE M1(',I,J,').'
          READ(*,*) M1(I,J)
70      CONTINUE
80    CONTINUE
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      DO 83 I=1,RANGE
        DO 81 J=1,RANGE
          WRITE(*,*) 'GIVE M2(',I,J,').'
          READ(*,*) M2(I,J)
81      CONTINUE
83    CONTINUE
      DO 110 I=1,RANGE
        DO 100 J=1,RANGE
          DO 90 K=1,RANGE
            NEW(J,K)=(M1(J,I)*M2(I,K))+NEW(J,K)
90        CONTINUE
100     CONTINUE
110   CONTINUE
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*) 'NEW IS:'
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      DO 120 I=1,RANGE
        WRITE(*,115) (NEW(I,J),J=1,RANGE)
115     FORMAT(' ',40(F8.5,1X))
120   CONTINUE
      STOP
      END
C
C
C
      PROGRAM INVER
C
C
C     This program calculates the inverse of an n x n matrix.
C
      INTEGER*2 K,RANGE,I,J,T,Q,SIZE,R,C,R1(40),C1(40),L,L1,L2,L3,L4,L5,
     X          L7
      REAL*8 DET,Y(40),A(40,40),M,E,B,B2,HOLD(40,40),P,AINV(40,40)
C
C
```

```
      WRITE(*,*) 'GIVE N.'
      READ(*,*) RANGE
      DO 2 I=1,RANGE
        DO 1 J=1,RANGE
          WRITE(*,*) 'GIVE M(',I,J,').'
          READ(*,*) A(I,J)
1       CONTINUE
2     CONTINUE
      DO 5 I=1,RANGE
        DO 4 J=1,RANGE
          HOLD(I,J)=A(I,J)
4       CONTINUE
5     CONTINUE
      K=0
      DET=1
20     K=K+1
      IF (K .GT. RANGE) THEN
        GOTO 47
      ENDIF
      I=0
      J=0
      B=0
22     I=I+1
      IF (I .GT. RANGE) THEN
        GOTO 34
      ENDIF
      Q=0
24     Q=Q+1
      IF (Q .GT. (K-1)) THEN
        GOTO 27
      ENDIF
25     IF (I .EQ. R1(Q)) THEN
        GOTO 22
      ENDIF
      GOTO 24
27     J=J+1
      IF (J .GT. RANGE) THEN
        J=0
        GOTO 22
      ENDIF
      Q=0
29     Q=Q+1
      IF (Q .GT. (K-1)) THEN
        GOTO 32
      ENDIF
      IF (J .EQ. C1(Q)) THEN
        GOTO 27
      ENDIF
      GOTO 29
32     E= ABS(A(I,J))
      IF (E .GE. B) THEN
        R1(K)=I
        C1(K)=J
```

```
        B=E
      ENDIF
      GOTO 27
34    L7=C1(K)
      L=R1(K)
      B2=ABS(A(L,L7))
      P=A(L,L7)
      IF (B2 .LE. 0.000000000001) THEN
        WRITE(*,*) 'DET = 0 AND INVERSE DOESN''T EXIST!!!!!!!'
        DET=DET*P
        DO 36 I=1,RANGE
          DO 35 J=1,RANGE
            A(I,J)=HOLD(I,J)
35        CONTINUE
36      CONTINUE
        GOTO 80
      ENDIF
      DET=DET*P
      R=R1(K)
      C=C1(K)
      J=0
38    J=J+1
      IF (J .GT. RANGE) THEN
        GOTO 41
      ENDIF
      IF (J .EQ. C) THEN
        A(R,C)=1.0/P
        GOTO 38
      ENDIF
      A(R,J)=A(R,J)/P
      GOTO 38
41    I=0
      J=0
42    I=I+1
      IF (I .GT. RANGE) THEN
        GOTO 20
      ENDIF
      M=A(I,C)
      IF (I .EQ. R) THEN
        GOTO 42
      ENDIF
44    J=J+1
      IF (J .GT. RANGE) THEN
        J=0
        GOTO 42
      ENDIF
      IF (J .EQ. C) THEN
        A(I,C)=M/(-P*1.0)
        GOTO 44
      ENDIF
      A(I,J)=A(I,J)-A(R,J)*M
      GOTO 44
47    I=0
```

```
    J=0
48   J=J+1
    IF (J .GT. RANGE) THEN
      I=0
      J=0
      GOTO 53
    ENDIF
49   I=I+1
    IF (I .GT. RANGE) THEN
      I=0
      GOTO 51
    ENDIF
    L1=C1(I)
    L2=R1(I)
    Y(L1)=A(L2,J)
    GOTO 49
51   I=I+1
    IF (I .GT. RANGE) THEN
      I=0
      GOTO 48
    ENDIF
    A(I,J)=Y(I)
    GOTO 51
53   I=I+1
    IF (I .GT. RANGE) THEN
      GOTO 58
    ENDIF
54   J=J+1
    IF (J .GT. RANGE) THEN
      J=0
      GOTO 56
    ENDIF
    L3=R1(J)
    L4=C1(J)
    Y(L3)=A(I,L4)
    GOTO 54
56   J=J+1
    IF (J .GT. RANGE) THEN
      J=0
      GOTO 53
    ENDIF
    A(I,J)=Y(J)
    GOTO 56
58   I=0
59   I=I+1
    IF (I .LE. RANGE) THEN
      L5=R1(I)
      Y(L5)=C1(I)
      GOTO 59
    ENDIF
    I=0
61   I=I+1
```

```
      IF (I .GT. RANGE) THEN
        DO 63 J=1,RANGE
          DO 62 T=1,RANGE
            AINV(J,T)=A(J,T)
            A(J,T)=HOLD(J,T)
62        CONTINUE
63      CONTINUE
        GOTO 67
      ENDIF
      J=0
64    J=J+1
      IF (J .GT. (RANGE-1)) THEN
        GOTO 61
      ENDIF
65    IF (Y(J) .LE. Y(J+1)) THEN
        GOTO 64
      ENDIF
      E=Y(J)
      Y(J)=Y(J+1)
      Y(J+1)=E
      DET=(-DET)
      GOTO 64
67    WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*) 'INVERSE IS:'
      WRITE(*,*)
      WRITE(*,*)
      DO 75 I=1,RANGE
        WRITE(*,70) (AINV(I,J),J=1,RANGE)
70      FORMAT(' ',40(F8.5,1X))
75    CONTINUE
80    STOP
      END
C
C
C
      PROGRAM PSQRT
C
C
C     This program calculates the positive square root of an n x n
C     matrix.
C
      INTEGER*2 I,J,K,MAX,N
      REAL*8 NUM,A(20,20),B(20,20),C(20,20),D(20,20),E(20,20)
      LOGICAL ERROR
C
C
      MAX=20
      NUM=0.0
      WRITE(*,*) 'GIVE N.'
      READ(*,*) N
```

```
      DO 2 I=1,N
        DO 1 J=1,N
          WRITE(*,*) 'GIVE M(',I,J,').'
          READ(*,*) D(I,J)
1       CONTINUE
2     CONTINUE
      DO 15 I=1,170
        CALL ASSIGN(B,A,N,MAX)
        DO 10 J=1,N
          DO 5 K=1,N
            IF (ABS(B(J,K)) .GT. 1.0D20) THEN
              ERROR=.TRUE.
              GOTO 20
            ENDIF
5         CONTINUE
10      CONTINUE
        CALL MATMUL(C,B,A,N,MAX)
        CALL MATSUB(E,D,C,N,MAX)
        NUM=0.5
        CALL MATSCA(C,NUM,E,N,MAX)
        CALL ASSIGN(B,C,N,MAX)
        CALL MATADD(C,A,B,N,MAX)
        CALL ASSIGN(A,C,N,MAX)
15    CONTINUE
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*) 'POSITIVE SQUARE ROOT IS:'
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)
      DO 17 I=1,N
        WRITE(*,16) (A(I,J),J=1,N)
16      FORMAT(' ',40(F8.5,1X))
17    CONTINUE
      WRITE(*,*)
      WRITE(*,*)
      GOTO 25
20    WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*) 'SQUARE ROOT DOESN''T EXIST.'
      WRITE(*,*)
25    STOP
      END
C
C
C
      SUBROUTINE MATSCA(NEW,SCALAR,M,RANGE,SIZE)
C
C     This subroutine calculates the product of a scalar and an n x n
C     matrix.
C
      INTEGER*2 SIZE,RANGE,I,J
```

```
      REAL*8 M(SIZE,1),NEW(SIZE,1),SCALAR
C
C
      DO 20 I=1,RANGE
        DO 10 J=1,RANGE
          NEW(I,J)=SCALAR*M(I,J)
10      CONTINUE
20    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE ASSIGN(NEW,M,RANGE,SIZE)
C
C     This subroutine assigns the values of one vector to another one.
C
      INTEGER*2 RANGE,SIZE,I,J
      REAL*8 M(SIZE,1),NEW(SIZE,1)
C
C
      DO 20 I=1,RANGE
        DO 10 J=1,RANGE
          NEW(I,J)=M(I,J)
10      CONTINUE
20    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE MATADD(NEW,M1,M2,RANGE,SIZE)
C
C     This subroutine calculates the sum of two n x n matrices.
C
      INTEGER*2 RANGE,SIZE,I,J
      REAL*8 M1(SIZE,1),M2(SIZE,1),NEW(SIZE,1)
C
C
      DO 20 I=1,RANGE
        DO 10 J=1,RANGE
          NEW(I,J)=M1(I,J)+M2(I,J)
10      CONTINUE
20    CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE MATMUL(NEW,M1,M2,RANGE,SIZE)
C
C     This subroutine calculates the product of two n x n matrices.
C
```

```
C
      INTEGER*2 I,J,K,SIZE,RANGE
      REAL*8 M1(SIZE,1),M2(SIZE,1),NEW(SIZE,1)
C
C
      DO 80 I=1,RANGE
        DO 70 J=1,RANGE
          NEW(I,J)=0.00
70      CONTINUE
80    CONTINUE
      DO 110 I=1,RANGE
        DO 100 J=1,RANGE
          DO 90 K=1,RANGE
            NEW(J,K)=(M1(J,I)*M2(I,K))+NEW(J,K)
90        CONTINUE
100     CONTINUE
110   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE MATSUB(NEW,M1,M2,RANGE,SIZE)
C
C     This subroutine calculates the difference between two n x n
C     matrices.
C
      INTEGER*2 RANGE,SIZE,I,J
      REAL*8 M1(SIZE,1),M2(SIZE,1),NEW(SIZE,1)
C
C
      DO 20 I=1,RANGE
        DO 10 J=1,RANGE
          NEW(I,J)=M1(I,J)-M2(I,J)
10      CONTINUE
20    CONTINUE
      RETURN
      END
```