THEORETICAL PREDICTION
OF VIBRATIONAL CIRCULAR DICHLORISM SPECTRA
OF R-GLYCYERALDEHYDE, R-ERYTHROSE, AND R-THREOSE

II. DEVELOPMENT OF A PROCEDURE TO SCALE
THE FORCE CONSTANT MATRIX EXPRESSED IN TERMS
OF INTERNAL COORDINATES

Daniel Zeroka
LEHIGH UNIVERSITY
Bethlehem, PA 18015-3172

James O. Jensen
RESEARCH DIRECTORATE

Janet L. Jensen
U.S. ARMY INFORMATION SYSTEMS COMMAND

November 1993

Approved for public release; distribution is unlimited.

Aberdeen Proving Ground, MD 21010-5423
Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorizing documents.
Theoretical Prediction of Vibrational Circular Dichroism Spectra of R-Glyceraldehyde, R-Erythrose, and R-Threose

Zeroka, Daniel (Lehigh University); Jensen, James O.; and Jensen, Janet L. (ERDEC)

Lehigh University, Department of Chemistry, Bethlehem, PA 18015-3172

DIR, ERDEC, *ATTN: SCBRD-RTE, APG, MD 21010-5423

A very important objective of the Detection Directorate at the U.S. Army Edgewood Research, Development and Engineering Center* is the remote detection of biological materials in the field. One line of thinking, currently being followed, is the recognition that sugars are distinguishing features of biological materials. In Part I of this study, the theoretical prediction of the vibrational circular dichroism (VCD) of the 3 and 4 carbon sugars - R-glyceraldehyde, R-erythrose, and R-threose is considered. The calculational procedure used involves determination of the frequencies corresponding to the normal modes of vibration. Since calculated frequencies at the Hartree-Fock level are typically 10% too high, some form of scaling of the frequencies or the force constant matrix is required for quantitative agreement with experimental measurements. In Part II of this study, a scaling method is described, and three key FORTRAN computer programs are presented. Basically, the force constant matrix in internal coordinates at the 6-31G* HF level of calculation is scaled to the calculated 6-31G* MP2 level of calculation. The force constant matrix in terms of Cartesian coordinates can be determined from a matrix transformation, originally shown by Pulay, involving the force constant matrix in terms of internal coordinates. The scaling constant for each off-diagonal element of the force constant matrix was determined by using the geometric mean $Q_i = (Q_{ij} Q_{ji})^{1/2}$ of the diagonal scaling constants $Q_i$ and $Q_j$. 

Approved for public release; distribution is unlimited.
4. TITLE AND SUBTITLE (continued)

II. Development of a Procedure to Scale the Force Constant Matrix Expressed in Terms of Internal Coordinates

11. SUPPLEMENTARY NOTES (Continued)

*When this study was conducted, ERDEC was known as the U.S. Army Chemical Research, Development and Engineering Center, and the ERDEC authors were assigned to the Research Directorate and U.S. Army Information Systems Command, respectively.
PREFACE

The work described in this report was authorized under Project No. 1O162622A553C, Reconnaissance, Detection, and Identification, Project No. 1O161102A71A, Research in CW/CB Defense, Contract No. DAAL03-91-C-0034, and Delivery Order No. 181. This work was started in May 1992 and completed in October 1992.

The use of trade names or manufacturers' names in this report does not constitute an official endorsement of any commercial products. This report may not be cited for purposes of advertisement.

This report has been approved for release to the public. Registered users should request additional copies from the Defense Technical Information Center; unregistered users should direct such requests to the National Technical Information Service.

Acknowledgments

This work was supported by the U.S. Army Edgewood Research, Development and Engineering Center (ERDEC)* (Dr. James O. Jensen) under the auspices of the U.S. Army Research Office Scientific Services Program administered by Battelle.

*When this study was conducted, ERDEC was known as the U.S. Army Chemical Research, Development and Engineering Center, and the ERDEC authors were assigned to the Research Directorate and U.S. Army Information Systems Command, respectively.
## CONTENTS

<table>
<thead>
<tr>
<th>II.1.</th>
<th>SCALING PROCEDURES</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>II.2.</td>
<td>SCALING PROGRAMS</td>
<td>2</td>
</tr>
<tr>
<td>II.2.1</td>
<td>Program bmat.f</td>
<td>2</td>
</tr>
<tr>
<td>II.2.2</td>
<td>Program matmult.f</td>
<td>14</td>
</tr>
<tr>
<td>II.2.3</td>
<td>Program matmult2.f</td>
<td>24</td>
</tr>
<tr>
<td>II.2.4</td>
<td>Program simplex.f</td>
<td>34</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>35</td>
</tr>
</tbody>
</table>
THEORETICAL PREDICTION
OF VIBRATIONAL CIRCULAR DICHROISM SPECTRA
OF R-GLYCERALDEHYDE, R-ERYTHROSE, AND R-THREOSE

II. DEVELOPMENT OF A PROCEDURE TO SCALE
THE FORCE CONSTANT MATRIX EXPRESSED IN TERMS
OF INTERNAL COORDINATES

II.1 SCALING PROCEDURES

In Part II of this report the procedure that was used in Part I to scale the force constant matrix is developed. More specifically the scaling methods used are discussed and the programs to implement the scaling are described and given. The harmonic force constant matrix gives the second derivative of the energy with respect to the coordinates of the molecule. The force constant can be expressed in two common ways. First, the force constant matrix can be expressed in terms of Cartesian coordinates as

\[ K_{ij} = \frac{\partial^2 E}{\partial q_i \partial q_j} \]  

where \( q_i \) is a Cartesian coordinate \( q_1 = x_1, q_2 = y_1, q_3 = z_1, \ldots, q_{3n} = x_n, q_{3n+1} = y_n, \) and \( q_{3n+2} = z_n, \) where \( n \) is the number of atoms in the molecule. The matrix \( K \) is \( 3n \times 3n. \) Second, the force constant matrix can be expressed in terms of \( 3n-6 \) internal coordinates, \( R, \) expressed as bond stretches, bond angle bends, dihedral angle torsion or other modes of motion. This force constant matrix is

\[ F_{ij} = \frac{\partial^2 E}{\partial R_i \partial R_j} \]  

The matrix \( F \) is \( (3n-6) \times (3n-6). \) There is a transformation between the internal and Cartesian coordinates given by

\[ R = Bq \]  

where \( R \) and \( q \) are column vectors whose components are the internal and Cartesian coordinates. Note that \( B \) is \( (3n-6) \times (3n). \) An existing computer program [1] to determine \( B \) given \( R \) and \( q \) was modified and named bmats.f. The program is given in the program section II.2; in addition, in Table 1 a sample setup of a datafile for \( R \)-glyceraldehyde, for use with bmats.f, is shown.

The following relation [2,3], originally shown by Pulay, between \( K \) and \( F \) holds

\[ F = B^{-1} K B^{-1} - \sum_i \phi_i B^{-1} C^i B^{-1} \]  

where \( \phi_i \) is the column vector of the forces expressed in internal coordinates, \( C^i \) is the second-order transformation matrix relating the Cartesian and internal coordinates, \( B^{-1} \) is the transpose of \( B^T \) and \( B^{-1} \) is given by

\[ B^{-1} = (B^T m B^{-1})^{-1} B^T \]  

where \( m \) is any matrix for which \((B^T m B^{-1})\) is not singular. [In the examples studied in this report, nonsingular matrices are obtained if \( m \) is taken to be the identity matrix.] We have
considered the force constant matrices at an optimized geometry; under that condition equation 4 becomes
\[ F = B^{-1} K B^{-1}. \] (6)

A program named matmul.f was written to carry out the matrix multiplication given by the previous equations. As a check on the program both Gaussian 90 and Gaussian 92 calculations on optimized geometries were run with option FREQ. This procedure will generate both force constant matrices K and F. The results of using the above matrix multiplication, for the examples considered, all agree exactly with the results obtained from the Gaussian calculations. The program matmul.f is given in section II.2.

Next, the FORTRAN program matmul.f was modified to allow for scaling of the force constant matrix, F. This new program is called matmul2.f. The scaling constants Q, are input into the program by editing matmul2.f. The resulting scaled F matrix is converted to a scaled K matrix which is then used as input to the CADPAC program to carry out a VCD calculation of allowed frequencies of vibrations and corresponding rotational strengths.

II.2 SCALING PROGRAMS

In this section 3 FORTRAN programs are reported.

II.2.1 Program bmat.f

The first program bmat.f determines the B matrix in the transformation between the internal R and Cartesian coordinates q where R is a (3n-6)-column vector and q is a (3n)-column vector. The data file used must be set for each molecule that is considered. Table 1 lists a sample data file for R-glyceraldehyde.

On the following pages a listing of the FORTRAN program bmat.f is given.
bmnt.f

.nf
C=342 GEN VIB ANAL PGM USING WILSON GF MATRIX METHOD
C
C THIS IS PROGRAM NUMBER 1 OF THE COMPLETE VIBRATIONAL PACKAGE.
C
C BMAT ... WILSON B MATRIX ELEMENTS FOR INTERNAL COORDINATES
C (VERSION 0 JUL 28, 1977)
C
C AUTHORS: MIKE PETERSON AND DOUG MCINTOSH, U OF T CHEM DEPT, TORONTO
C
C INPUT:
C
2 TITLE CARDS (20A4)
C
NOAT, IPNCHB (2I4)
NOAT0 NUMBER OF ATOMS (<=20).
IPNCHB0 PUNCH B MATRIX IF NON-ZERO (SEE NOTE BELOW).
C
X,Y,Z,ID (3G12.6,11A4)
X,Y,Z CARTESIAN COORDINATES OF AN ATOM.
ID0 FREE FORMAT LABEL (COLS 37-80).
REPEAT NOAT TIMES.
C
ICODE, I,J,K,L,IX,JX, FACTOR, ID (7I4,G12.6,1OA4)
ICODE0 INTERNAL COORDINATE TYPE (SEE BELOW). IF ICODE<0, THE
NEW B MATRIX ELEMENTS ARE ADDED TO THE PREVIOUS ONES.
I,J,K,L0 ATOM NUMBERS INVOLVED.
IX,JX0 OPTIONAL WEIGHTING OF INTERNAL COORD BY THE IX-JX BOND
LENGTH (NOT USED IF IX AND/OR JX IS 0).
FACTOR0 NEW ROW OF B IS MULTIPLIED BY FACTOR (BEFORE BEING
ADDED TO PREVIOUS ROW, IF ICODE<0). FACTOR DEFAULTS TO
1.0. USE TO COMBINE INTERNAL COORDINATES, IF DESIRED.
ID0 FREE FORMAT LABEL (COLS 41-80).
REPEAT AS OFTEN AS REQUIRED, TERMINATING WITH A BLANK CARD.
The TOTAL NUMBER OF INTERNAL COORDS MUST BE <= 3*NOAT.
C
ENTIRE DECK MAY BE REPEATED
C
ICODE MODE
C
1 BOND STRETCH
I AND J ARE ATOMS INVOLVED. K,L,IX,JX MUST BE 0.
C
2 VALENCE ANGLE BEND
I AND K ARE TERMINAL ATOMS, J IS CENTRAL ATOM. L MUST BE 0.
I,J,K MUST NOT BE COLINEAR.
C
3 OUT OF PLANE WAG
I IS WAGGED ATOM, J IS APEX ATOM, K AND L ARE ANCHOR ATOMS.
C
4 TORSION
J AND K DEFINE THE BOND UNDER TORSION. I AND L ARE THE NO OF
ATOMS (<=5) ATTACHED TO J AND K RESPECTIVELY. THE FOLLOWING 2
CARDS GIVE THE ATOM NOs FOR THE I-TYPE AND L-TYPE ATOMS (EACH
CARD IS 5I4). NONE OF THE I'S OR L'S SHOULD BE THE SAME, OR
EQUAL TO J OR K. THE TORSION IS PROPERLY NORMALIZED (SEE R L
HILDERBRANDT, J MOLEC SPEC, 44, 599 (1972)).
C
5 LINEAR BEND (DEFINES 2 INTERNAL COORDINATES)
C
6 LINEAR BEND (DEFINES 1 INTERNAL COORDINATE)
C
I AND K ARE END ATOMS, J IS CENTRAL ATOM. THE FOLLOWING CARD
GIVES A POINT (IN 3G12.6 FORMAT) PERPENDICULAR TO I-J-K AT J

-3-
WHICH ORIENTS THE BENDING COORDINATE. L MUST BE 0.

FOR ICODE=5 A PERPENDICULAR INTERNAL COORD IS ALSO DEFINED.

B IS (NOB,NA) WHERE NA=3*NOAT

B MUST BE DEFINED AS A SQUARE MATRIX FOR PROGRAMS 2 (F TRY/ATOM DISP) AND 3 (FORCE CONSTANT FITTER) OF THE VIBRATIONAL PACKAGE.

X0, X, Y, Z COORDINATES OF THE ATOMS (SIZEO (3,NOAT))

REQUIRED SUBROUTINES BOST, BEND, OPLA, TORS, LIBE

SUBROUTINES BOST, BEND, OPLA AND LIBE WERE MODIFIED FROM J H SCHACHT-SCHNEIDER'S 'GMAT' PROGRAM (SHELL DEVELOPMENT CO) WITH PERMISSION.

IMPLICIT REAL*8 (A-H,O-Z)

TO REDIMENSION, CHANGE FOLLOWING CARD AND ALL OTHER BLANK COMMON

COMMON IC,N1,N2,N3,N4,N5,N6,NOAT,NOB,IER,X(3,70),B(200,200)

INTEGER TITLE(40),IDC(11),IDQ(10)

READ TITLE CARDS

READ(5,1000,END=210)TITLE,NOAT,IPNCHB

WRITE(6,1010)TITLE,NOAT

NA=NOAT*3

DO 20 I=1,NOAT

READ(5,1020)(X(J,I),J=1,3),IDC

20 WRITE(6,1030)I,(X(J,I),J=1,3),IDC

NOB=0

IER=0

ISCAN IS 0 NORMALLY, >0 FOR ERROR SCAN AFTER AN INPUT ERROR IS FOUND

ISCAN=0

WRITE(6,1040)

READ INTERNAL COORD DEFINITIONS

READ(5,1050)ICODE,N1,N2,N3,N4,N5,N6,FACTOR,IDQ

IF (ICODE.EQ.0) GO TO 150

IF(ICODE.GT.0)NOB=NOB+1

IF(ICODE.EQ. -5)NOB=NOB-1

WRITE(6,1060)NOB,ICODE,N1,N2,N3,N4,N5,N6,FACTOR,IDQ

IF THIS IS A NEW COORDINATE, INCREMENT NOB

IF(ICODE.GT.0)NOB=NOB+1

IF(ICODE.EQ.-5)NOB=NOB+1

IF(NOB.GT.NA)GO TO 180

ZERO ROW OF B

DO 40 J=1,NA

40 B(NOB,J)=0.D0

IC=IABS(ICODE)

GO TO (1,2,3,4,5,6),IC

WRITE(6,1070)ICODE

GO TO 200

CALL BOST

GO TO 60

CALL BEND

GO TO 60

CALL OPLA

GO TO 60

CALL TORS

GO TO 60

ZERO EXTRA ROW OF B IF ICODE=+-5
5 I=NOB+1
  IF(I.GT.NA)GO TO 180
  DO 50 J=1,NA
50  B(I,J)=0.DO
  6 CALL LIBE
  60 IF(IER.NE.0)GO TO 190
C MULTIPLY NEW ROW(S) BY FACTOR (IF NOT 1.0)
  IF(FACTOR.EQ.1.DO)GO TO 105
  IF(IC.EQ.5)GO TO 80
  70 ISW=0
  I=NOB
  GO TO 90
  80 ISW=1
  I=NOB-1
  90 DO 100 J=1,NA
100  B(I,J)=B(I,J)*FACTOR
C DO WE ADD CURRENT ROW(S) TO PREVIOUS ROW(S) ?
  105 IF(ICODE.GT.0)GO TO 30
  IF(ICODE.EQ.-5)GO TO 110
  ISW=0
  I=NOB-1
  GO TO 130
  110 ISW=1
  120 I=NOB-2
  130 DO 140 J=1,NA
140  B(I,J)=B(I,J) + B(NOB,J)
  NOB=NOB-1
  IF(ISW.EQ.0)GO TO 30
  ISW=0
  GO TO 120
  150 IF(ISCAN.NE.0)GO TO 10
  WRITE(6,1080)NOB
  K=-11
  160 K=K+12
  L=MIN0(K+11,NA)
  WRITE(6,1090)(J,J=K,L)
  DO 170 I=I,NOB
    OPEN(22, FILE='BMAT.IN')
    WRITE(22,1101)(B(I,J),J=KL)
170  WRITE(6,1100)I,(B(IJ),J=K,L)
C..DZ DO 171 I=1,15
C..DZ WRITE(22,1101)(B(I,J),J=1,12)
C..DZ 171 CONTINUE
C..DZ DO 172 I=1,15
C..DZ WRITE(22,1101)(B(I,J),J=13,21)
C..DZ 172 CONTINUE
  IF(L.LT.NA)GO TO 160
C EACH ELEMENT OF B IS PUNCHED IN AS FORMAT - THE INTERNAL 64 BIT (8
C BYTE) FLOATING POINT NUMBER IS INTERPRETED AS 8 EBCDIC CHARACTERS (1
C CHARACTER IS STORED IN 1 BYTE (= 8 BITS) IN IBM 360/370 COMPUTERS).
C EACH DOUBLE PRECISION (REAL*8) VALUE THEN OCCUPIES 8 CARD COLUMNS -
C THIS FORMAT MINIMIZES THE SIZE OF THE B MATRIX CARD DECK, BUT IS
C THEN COMPLETELY INCOMPREHENSIBLE. DO NOT INTERPRET THESE CARDS.
  IF(IPNCHB.NE.0)WRITE(7,1160)TITLE,NOBNA,((B(I,J),I=1,NOB),J=1,NA)
  GO TO 10
  180 WRITE(6,1140)
  STOP
  190 IF(IER.EQ.1)WRITE(6,1130)
  200 IF(ISCAN.EQ.0)WRITE(6,1170)
C ERROR SCAN FOR THIS DATA DECK, AND DON'T PRINT/PUNCH B MATRIX
ISCAN=ISCAN+1
IER=0
GO TO 30
210 WRITE(6,1120)
STOP
1000 FORMAT(20A4/20A4/2I4)
1010 FORMAT('I',20A4,24X,’BMAT (VERSIONO JUL 28, 1977)'/1X,20A4/
1020 FORMAT(3G12.6,11A4)
1030 FORMAT(’0’,I3,2X,3F12.6,5X,11A4)
1040 FORMAT(‘0 INTERNAL COORDINATE DEFINITIONS0’/’0NOB CODE I ’,
$ ’J K L IX JX FACTOR’)
1050 FORMAT(7I4,G12.6,10A4)
1060 FORMAT(1X,13,2I4,5I4,F11.6,1X,10A4)
1070 FORMAT( ’0ILLEGAL CODE’,I5,’ CHosen’)
1080 FORMAT( ‘ONUMBER OF INTERNAL COORDINATES =’,I4/’1B MATRIX ’,
$ ’(NOB BY 3*NOAT)0’)
1090 FORMAT(’0’,12I10)
1100 FORMAT(’0’,I4,2X,12F10.6)
C1101 FORMAT(I4,2X,12E15.6)
1101 FORMAT(12E15.6)
1120 FORMAT(’1*** NORMAL TERMINATION’/)
1130 FORMAT(’ILLEGAL SPECIFICATION OF I, J, K, L, IX OR JX’)
1140 FORMAT(’0*** PROGRAM TERMINATED - TOO MANY INTERNAL COORDS’/)
C1160 FORMAT(20A4/20A4/2I4/(10A8))
1160 FORMAT(20A4/20A4/2I4/(3D23.16))
1170 FORMAT(’0*** PROGRAM WILL SCAN FOR FURTHER ERRORS IN DATA DECK’/)
END-
SUBROUTINE BOST
C THIS SUBROUTINE COMPUTES THE B MATRIX ELEMENTS FOR A BOND STRETCH
C AS DEFINED BY WILSON.
IMPLICIT REAL*8 (A-H,O-Z)
COMMON IC,I,J,K,L,IX,JX,NOAT,NOB,IER,X(3,70),B(200,200)
COMMON/SCHACH/RRIJ(3),RJL(3),EIJ(3),EJK(3),EKL(3)
IF(I.LE.0.OR.I.GT.NOAT)GO TO 30
IF(J.LE.0.OR.J.GT.NOAT)GO TO 30
IF(K.NE.0)GO TO 30
IF(L.NE.0)GO TO 30
IF(IX.NE.0)GO TO 30
IF(JX.NE.0)GO TO 30
DIJSQ=0.0
DO 10 M=1,3
T=X(M,J)-X(M,I)
RIJ(M)=T
10 DIJSQ=DIJSQ+T*T
DIJ=DJSQRT(DIJSQ)
II=3*(I-1)
JJ=3*(J-1)
DO 20 M=1,3
T=RIJ(M)
IF(DABS(T).LT.1.D-8)GO TO 20
T=T/DIJ
B(NOB,II+M)=-T
B(NOB,JJ+M)=T
20 CONTINUE
RETURN
30 IER=1
RETURN
END-
SUBROUTINE BEND
C THIS SUBROUTINE COMPUTES THE B MATRIX ELEMENTS OF A VALENCE
C ANGLE BENDING COORDINATE AS DEFINED BY WILSON.
C I AND K ARE THE NUMBERS OF THE END ATOMS.
C J IS THE NUMBER OF THE CENTRAL ATOM.
IMPLICIT REAL*8 (A-H,O-Z)
COMMON IC, I, J, K, L, IX, JX, NOAT, NOB, IER, X(3,70), B(200,200)
COMMON/SCHACH/RJI(3), RJK(3), RXL(3), EJI(3), EJK(3), EKL(3)
IF(I.LE.0.OR.I.GT.NOAT)GO TO 50
IF(J.LE.0.OR.J.GT.NOAT)GO TO 50
IF(K.LE.0.OR.K.GT.NOAT)GO TO 50
IF(L.NE.0)GO TO 50
IF(IX.LT.0.OR.IX.GT.NOAT)GO TO 50
IF(JX.LT.0.OR.JX.GT.NOAT)GO TO 50
IF(IX.NE.0.AND.JX.NE.0)GO TO 10
IX=1
JX=1
10 DJISQ=0.DO
DJKSQ=0.DO
DXSQ=0.DO
DO 20 M=1,3
TP=X(M,J)
T=X(M,I)-TP
RJI(M)=T
DJISQ=DJISQ+T*T
T=X(M,K)-TP
RJK(M)=T
DJKSQ=DJKSQ+T*T
T=X(M,JX)-X(M,IX)
20 DXSQ=DXSQ+T*T
DJJ=DJSQRT(DJISQ)
DJK=DSQRT(DJKSQ)
DX=DSQRT(DXSQ)
IF(DX.EQ.0.DO)DX=1.DO
DOTJ=0.DO
DO 30 M=1,3
T=RJI(M)/DJJ
EJI(M)=T
TP=RJK(M)/DJK
EJK(M)=TP
30 DOTJ=DOTJ+T*TP
IF(ABS(DOTJ).GT.0.99995DO)GO TO 60
SINJ=DSQRT(1.DO-DOTJ*DOTJ)
II=3*(I-1)
JJ=3*(J-1)
KK=3*(K-1)
DO 40 M=1,3
SMI=DX*(DOTJ*EJ(M)-EJK(M))/((DJJ*SINJ)
IF(ABS(SMI).GE.1.DO)B(NOB,II+M)=SMI
SMK=DX*(DOTJ*EJK(M)-EJI(M))/((DJK*SINJ)
IF(ABS(SMK).GE.1.DO)B(NOB,KK+M)=SMK
SUM=SMI+SMK
40 IF(ABS(SUM).GE.1.DO)B(NOB,JJ+M)=-SUM
RETURN
50 IER=1
RETURN
60 IER=-1
WRITE(6,1000)
RETURN
1000 FORMAT('I-J-K IS COLINEAR - USE LINEAR BEND')
END

SUBROUTINE OPLA
C THIS SUBROUTINE COMPUTES THE B MATRIX ELEMENTS FOR AN OUT OF
C PLANE WAGGING COORDINATE AS DEFINED BY DECIUS, MCINTOSH, MICHAELIAN
C AND PETERSON. SUBROUTINE CODED BY M PETERSON, UNIV OF TORONTO.
C I IS THE END ATOM (ATOM WAGGED WITH RESPECT TO J-K-L PLANE).
C J IS THE APEX ATOM (ATOMS I, K AND L ARE ATTACHED TO J).
IMPLICIT REAL*8 (A-H,O-Z)
COMMON IC, I, J, K, L, IX, JX, NOAT, NOB, IER, X(3, 70), B(200, 200)
COMMON/SCHACH/RJI(3), RJK(3), RJL(3), EJI(3), EJK(3), EJL(3)
DIMENSION C1(3)
IF(I.LE.0.OR.I.GT.NOAT)GO TO 60
IF(J.LE.0.OR.J.GT.NOAT)GO TO 60
IF(K.LE.0.OR.K.GT.NOAT)GO TO 60
IF(L.LE.0.OR.L.GT.NOAT)GO TO 60
IF(IX.LT.0.OR.IX.GT.NOAT)GO TO 60
IF(JX.LT.0.OR.JX.GT.NOAT)GO TO 60
IF(IX.NE.0.AND.JX.NE.0)GO TO 10
IX=1
JX=1
10 DJISQ=0.D0
DJKSQ=0.D0
DJLSQ=0.D0
DXSQ=0.D0
DO 20 M=1,3
TP=X(M, J)
T=X(M, I) -TP
RJI(M)=T
DJISQ=DJISQ+T*T
T=X(M, K) -TP
RJK(M)=T
DJKSQ=DJKSQ+T*T
T=X(M, L) -X(M, J)
RJL(M)=T
DJLSQ=DJLSQ+T*T
T=X(M, JX) -X(M, IX)
20 DXSQ=DXSQ+T*T

DO 30 M=1,3
T=RJI(M)/DJ
EJI(M)=T
TP=RJK(M)/DJ
EJK(M)=TP
TPP=RJL(M)/DJ
EJL(M)=TPP
COSI=COSI+TP*TPP
COSK=COSK+T*TPP
30 COSL=COSL+T*TPP

IF(DABS(COSI).GT.0.99995D0)GO TO 70
SINSIN=1.D0-COSI*COSI
SINI=DSQRT(SINSIN)
C1(1)=EJK(2)*EJL(3)-EJK(3)*EJL(2)
SUBROUTINE TORS

C THIS SUBROUTINE COMPUTES THE B MATRIX ELEMENTS FOR TORSION AS DEFINED
C BY R L HILDERBRANDT IN J MOLEC SPEC, 44, 599 (1972).
C SUBROUTINE CODED BY M PETERSON, DEPT OF CHEMISTRY, UNIV OF TORONTO.
C J AND K DEFINE THE BOND UNDER TORSION.
C NI AND NL ARE THE NUMBER OF ATOMS ATTACHED TO J AND K RESPECTIVELY
C (NI, NL <= 5). 2 DATA CARDS ARE READ0 (1) CONTAINS NI ATOM NUMBERS
C FOR THE I-TYPE ATOMS, AND (2) CONTAINS NL ATOM NUMBERS FOR THE L-TYPE
C ATOMS (BOTH CARDS ARE IN 514 FORMAT).
C
C IATOM, LATOM0 ATOM NUMBERS FOR THE I- AND L-TYPE ATOMS (SIZE0 5)
IMPLICIT REAL*(A-H,O-Z)
COMMON IC,N1,J,K,NL,IX,JX,NOAT,IER,X(3,70),B(200,200)
COMMON/SCHACH/RIJ(3) ,RJK(3) ,RLK(3) ,EIJ(3) ,EJK(3) ,ELK(3)
DIMENSION CR(3),IATOM(5),LATOM(5),SJ(3),SK(3)
READ(5,1000) (IATOM(I) ,I=1,NI)
WRITE(6,10100) (IATOM(I) ,I=1,NI)
READ(5,1000) (LATOM(L) ,L=1,NL)
WRITE(6,10200) (LATOM(L) ,L=1,NL)
IF(NI.LE.0.OR.NI.GT.5)GO TO 110
IF(J.LE.0.OR.J.GT.NOAT)GO TO 110
IF(K.LE.0.OR.K.GT.NOAT)GO TO 110
IF(NL.LE.0.OR.NL.GT.5)GO TO 110

C1(2)=EJK(3)*EJL(1)-EJK(1)*EJL(3)
C1(3)=EJK(1)*EJL(2)-EJK(2)*EJL(1)
DOT=EJI(1)*C1(1)+EJI(2)*C1(2)+EJI(3)*C1(3)
SINT=DOT/SINI
IF(DABS(SINT) .GT.0.00005D0)WRITE(6,1020)
IF(DABS(SINT) .GT.0.99995D0)GO TO 80
COST=DSQRT(1.0D0-SINT*SINT)
TANT=SINT/COST
II=3*(I-1)
JJ=3*(J-1)
KK=3*(K-1)
LL=3*(L-1)
COST=SINT/COST
DO 50 M=1,3
T=C1(M)/COST
SMI=(T-TANT*EJI(14))/DJI
IF(DABS(SMI) .GE.1.D-8)B(NOB,1140)=DX*SMI
SMI=T*COSI*COSK-COSL)/(SINSIN*DJK)
IF(DABS(SMI) .GE.1.D-8)B(NOB,KK+M)=DX*SMI
SMI=T*COSI*COSL-COSK)/(SINSIN*DKJ)
IF(DABS(SMI) .GE.1.D-8)B(NOB,LL+M)=DX*SMI
SUM=SMI+SMI+SMI

1000 FORMAT( ' K-J-L IS COLINEAR (NO PLANE DEFINED FOR WAG OF I)'
1010 FORMAT( ' I IS PERPENDICULAR TO J-K-L PLANE - USE VALENCE ANGLE ' ,
$ ' BENDS')
1020 FORMAT( '+I,86X,*** WARNING WAG OF A NON-PLANAR SYSTEM ****')
END

SUBROUTINE TORS
IF(IX.LT.0. OR IX.GT.NOAT) GO TO 110
IF(JX.LT.0. OR JX.GT.NOAT) GO TO 110
IF(IX.NE.0. AND JX.NE.0) GO TO 10
IX=1
JX=1
10 DJKSQ=0.D0
DXSQ=0.D0
DO 20 M=1,3
SJ(M)=0.D0
SK(M)=0.D0
T=X(M,K)-X(M,J)
RJK(M)=T
DJKSQ=DJKSQ+T*T
T=X(M,JX)-X(M,IX)
20 DXSQ=DXSQ+T*T
DJK=1.D0/DSQRT(DJKSQ)
DX=DSQRT(DXSQ)
IF(DX.EQ.0.D0)DX=1.D0
DO 30 M=1,3
30 EJK(M)=RJK(M)*DJK
JJ=3*(J-1)
KK=3*(K-1)
C LOOP OVER THE I-TYPE ATOMS
DO 60 N=1,NI
I=IATOM(N)
IF(I.LE.0. OR I.GT.NOAT) GO TO 110
DIJSQ=0.D0
DO 40 M=1,3
T=X(M,J)-X(M,I)
RIJ(M)=T
40 DIJSQ=DIJSQ+T*T
DIJ=1.D0/DSQRT(DIJSQ)
COSJ=0.D0
DO 50 M=1,3
T=RJ(M)*DIJ
EIJ(M)=T
50 COSJ=COSJ-T*EJK(M)
IF(DABS(COSJ).GT.0.99995D0) GO TO 120
SIN2J=(1.D0-COSJ*COSJ)*DFLOAT(NI)
II=3*(I-1)
CR(1)=EIJ(2)*EJK(3)-EIJ(3)*EJK(2)
CR(2)=EIJ(3)*EJK(1)-EIJ(1)*EJK(3)
CR(3)=EIJ(1)*EJK(2)-EIJ(2)*EJK(1)
DO 60 M=1,3
T=CR(M)/SIN2J
SMI=T*DIJ
IF(DABS(SMI).GE.1.D-8)B(NOB,II+M)=-DX*SMI
SMK=T*COSJ*DJK
SK(M)=SK(M)+SMK
SMJ=SMI-SMK
60 SJ(M)=SJ(M)+SMJ
C LOOP OVER THE L-TYPE ATOMS
DO 90 N=1,NL
L=LATOM(N)
IF(L.LE.0. OR L.GT.NOAT) GO TO 110
DLKSQ=0.D0
DO 70 M=1,3
T=X(M,K)-X(M,L)
RLK(M)=T
70 DLKSQ=DLKSQ+T*T
DLK=1.D0/DSQRT(DLKSQ)
COSK=0.D0
DO 80 M=1,3
T=RLK(M)*DLK
ELK(M)=T
80 COSK=COSK+EJK(M)*T
IF(DABS(COSK).GT.0.99999D0)GO TO 120
SIN2K=(1.D0-COSK*COSK)*DFLOAT(NL)
LL=3*(L-1)
CR(1)=ELK(3)*EJK(2)-ELK(2)*EJK(3)
CR(2)=ELK(1)*EJK(3)-ELK(3)*EJK(1)
CR(3)=ELK(2)*EJK(1)-ELK(1)*EJK(2)
DO 90 M=1,3
T=CR(M)/SIN2K
SML=T*DLK
IF(DABS(SML).GE.1.D-6)B(NOB,LL+M)=-DX*SML
SMJ=T*COSK*DJK
SJ(M)=SJ(M)+SMJ
SMK=SML-SMJ
90 SK(M)=SK(M)+SMK
DO 100 M=1,3
SMJ=SMJ
IF(DABS(SMJ).GE.1.D-8)B(NOB,JJ+M)=SMJ*DX
SMK=SK(M)
100 IF(DABS(SMK).GE.1.D-8)B(NOB,KK+M)=SMK*DX
RETURN
110 IER=1
RETURN
120 IER=-1
WRITE(6,1030)
RETURN
1000 FORMAT(5I4)
1010 FORMAT('+-',86X,'I0',5I4)
1020 FORMAT('+-',109X,'L0',5I4)
1030 FORMAT('I-J-K OR J-K-L IS COLINEAR (NO TORSION POSSIBLE')
END

SUBROUTINE LIBE
C THIS SUBROUTINE COMPUTES THE B MATRIX ELEMENTS FOR A LINEAR BEND
C OR FOR A PAIR OF PERPENDICULAR LINEAR BENDS.
C I AND K ARE THE END ATOMS.
C J IS THE CENTRAL ATOM.
C
C A GIVES THE CARTESIAN COORDINATES OF A POINT IN SPACE, SUCH
C THAT THE VECTOR FROM ATOM J TO POINT A IS PERPENDICULAR TO
C THE LINE I-J-K AND SERVES TO ORIENT THE COORDINATES IN SPACE.
IMPLICIT REAL*8 (A-H,O-Z)
COMMON IC,I,J,K,L,IX,JX,NOAT,NOB,IER,X(3,70),B(200,200)
COMMON/SCHACH/RJI(3),RJK(3),EJK(3),UP(3),UN(3),UNIT(3)
DIMENSION A(3)
READ(5,1000)A
WRITE(6,1010)A
IF(I.LE.0.OR.I.GT.NOAT)GO TO 60
IF(J.LE.0.OR.J.GT.NOAT)GO TO 60
IF(K.LE.0.OR.K.GT.NOAT)GO TO 60
IF(L.LE.0.)GO TO 60
IF(IX.LT.0.OR.IX.GT.NOAT)GO TO 60
IF(JX.LT.0.OR.JX.GT.NOAT)GO TO 60
IF(IX.NE.0.AND.JX.NE.0)GO TO 10
IX=1
JX=1
10 DJISQ=0. D0
DJKSQ=0. D0
DXSQ=0. D0
DJASQ=0. D0
DO 20 M=1,3
TP=X(M, J)
T=X(M, I)-TP
RJI(M)=T
DJISQ=DJISQ+T*T
T=X(M, K)-TP
RJK(M)=T
DJKSQ=DJKSQ+T*T
T=X(M, JX)-X(M, IX)
DXSQ=DXSQ+T*T
T=A(M)-TP
UN(M)=T
20 DJASQ=DJASQ+T*T
DJI=DSQRT(DJISQ)
DJK=DSQRT(DJKSQ)
DX=DSQRT(DXSQ)
DJA=DSQRT(DJASQ)
IF(DX.EQ.0.D0)DX=1.D0
DOTJ=0.D0
DOTP=0.D0
DO 30 M=1,3
T=RJI(M)/DJI
TP=RJK(M)/DJK
EJK(M)=TP
DOTJ=DOTJ+T*TP
TP=UN(M)/DJA
UNIT(M)=TP
30 DOTP=DOTP+T*TP
TEST=ABS(DOTJ)-1.D0
IF(DABS(TEST).GT.0.00005D0)GO TO 70
IF(DABS(DOTP).GT.0.00005D0)GO TO 80
II=3*(I-1)
JJ=3*(J-1)
KK=3*(K-1)
DO 40 M=1,3
T=UNIT(M)
IF(DABS(T).LT.1.D-8)GO TO 40
T=-DX*T
SMI=T/DJI
B(NOB, II+M)=SMI
SMK=T/DJK
B(NOB, KK+M)=SMK
B(NOB, JJ+M)=-SMI-SMK
40 CONTINUE
IF(IC.EQ.6)RETURN
NOB=NOB+1
UP(1)=EJK(2)*UNIT(3)-EJK(3)*UNIT(2)
UP(2)=EJK(3)*UNIT(1)-EJK(1)*UNIT(3)
UP(3)=EJK(1)*UNIT(2)-EJK(2)*UNIT(1)
DO 50 M=1,3
T=UP(M)
IF(DABS(T).LT.1.D-8)GO TO 50
T=-DX*T
SMI=T/DJI
B(NOB, II+M)=SMI
SMK=T/DJK
B(NOB, KK+M) = SMK
B(NOB, JJ+M) = -SMI-SMK

50 CONTINUE
   RETURN
60 IER=1
   RETURN
70 IER=-1
   WRITE(6, 1020)
   RETURN
80 IER=-1
   WRITE(6, 1030)
   RETURN
1000 FORMAT(3G12.6)
1010 FORMAT(’,+’, 86X,’A = (’,2(F11.7, ’),F11.7,’))
1020 FORMAT(’ I-J-K NOT COLINEAR ~ USE VALENCE ANGLE BEND’)
1030 FORMAT(’ ATOM A NOT PERPENDICULAR TO I-J-K AT J’)
END
II.2.2 Program matmul.t

The next program is matmul.f. This program carries out the transformation

\[ F = B^+ K B^{-1} \]  

and also the determination of K from F. The parameter NAT in the program represents the number of atoms in the molecule considered and must be changed for each molecule considered. The file containing the matrix K, KMAT.IN, can be obtained from a Gaussian calculation or a CADPAC calculation.

On the following pages a listing of the FORTRAN program matmul.f is given.
PROGRAM MAIN
PARAMETER(NAT=16, MM=3*NAT-6, N=3*NAT, MM=2*MM, NROW=MM, +NMATR=NROW*(NROW+1)/2)
IMPLICIT REAL*8 (A-H, O-Z)
REAL*8 K(N,N), M(N,N), F1D(NMATR)
REAL MC, MO, MH
DIMENSION B(MM, N), BM(MM, N), BP(N, MM), F(MM, MM), TEST(MM, MM),
1 BMBP(MM, MM), BMBPI(MM, MM), BPLM(MM, N), PROD(MM, N), BPLMI(N, MM),
2 TESTA(MM, MM), AA(MM, MM), BB(MM, MM), TEST2(MM, MM)
COMMON NSYS, INDEX, DET

C... GET MATRICES B AND K
C...
CALL BKMATR(MM, N, B, K)
C...
ADJOINT OF B
C...
DO 10 I=1,N
DO 10 J=1,MM
   BP(I,J)=B(J,I)
10 CONTINUE
C...
DETERMINE PRODUCT OF B M BP MATRICES
C...
MC=12.01
MO=16.00
MH=1.008
DO 501 I=1,N
   DO 501 J=1,N
      M(I,J)=0.
   501 CONTINUE
502 CONTINUE
DO 11 I=1,MM
   DO 11 J=1,N
      SUM=0.
      DO 11 L=1,N
         SUM=SUM+B(I,L)*M(L,J)
      BM(I,J)=SUM
   11 CONTINUE
C...
DETERMINE PRODUCT OF B M BP MATRICES
DO 511 I=1,MM
   DO 511 J=1,MM
      SUM=0.
      DO 511 L=1,N
         SUM=SUM+BM(I,L)*BP(L,J)
      BMBP(I,J)=SUM
   511 CONTINUE
OPEN(3,FILE='TESTADZ.OUT')
OPEN(23,FILE='BMBP.mat')
WRITE(3,*) 'BMBP'
WRITE(3,115) (BMBP(I,J),J=1,MM)
WRITE(23,115) (BMBP(I,J),J=1,MM)
-15-
CONTINUE

DETERMINE INVERSE OF B M BP

DO 12 I=1,MM
DO 12 J=1,MM

AA(I,J)=BMP(I,J)

12 CONTINUE

NSYS=0
INDEX=1
DO 221 I=1,MM
DO 221 J=MM+1,MM+MM

AA(I,J)=0.


221 CONTINUE

DO 222 I=1,MM
DO 222 J=1,MM

BB(I,J)=0.

IF(I.EQ.J) BB(I,J)=1.

222 CONTINUE

CALL MATCALC(AA, BB, MM, MM+MM)
WRITE(6,*), 'DETA =', DET

SET BMBPI MATRIX

DO 191 I=1,MM
DO 191 J=1,MM

BMBPI(I,J)=AA(I,J+MM)

191 CONTINUE

WRITE(3,*), 'BMBPI'
DO 192 I=1,MM

WRITE(3,116) (BMBPI(I,J), J=1,MM)

192 CONTINUE

DETERMINE TESTA MATRIX

DO 302 I=1,MM
DO 302 J=1,MM

SUM=0.
DO 302 L=1,MM

SUM=SUM+BMBPI(I,L)*BMP(I,J)

TESTA(I,J)=SUM

302 CONTINUE

WRITE(3,*), 'TESTA'
DO 340 I=1,MM

WRITE(3,111) (TESTA(I,J), J=1,MM)

340 CONTINUE

DETERMINE BPLM MATRIX

DO 13 I=1,MM
DO 13 J=1,N

SUM=0.
DO 13 L=1,MM

SUM=SUM+BMBPI(I,L)*BM(I,J)

BPLM(I,J)=SUM

13 CONTINUE
DETERMINE BPLM * BP MATRIX

DO 202 I=1,MM
   DO 202 J=1,MM
      SUM=0.
      DO 202 L=1,N
         SUM=SUM+BPLM(I,L)*BP(L,J)
      END
      TEST(I,J)=SUM
   END
   CONTINUE
   OPEN(2,FILE='TESTDZ.OUT')
   WRITE(2,'*') 'BPLM * BP'
   DO 240 I=1,6
      WRITE(2,111) (TEST(I,J),J=1,6)
   END
   CONTINUE
   WRITE(2,102)
   DO 241 I=1,15
      WRITE(2,112) (TEST(I,J),J=13,15)
   END
   CONTINUE

DETERMINE TRANSPOSE OF BPLM MATRIX

DO 14 I=1,N
   DO 14 J=1,MM
      BPLMI(I,J)=BPLM(J,I)
   END
   CONTINUE

WRITE(2,*)'K MATRIX

DO 43 I=1,15
   WRITE(2,111) (K(I,J),J=1,12)
END
CONTINUE
   WRITE(2,102)
   DO 44 I=1,15
      WRITE(2,112) (K(I,J),J=13,15)
   END
   CONTINUE

DETERMINE PRODUCT OF BPLM K BPLMI MATRICES TO GIVE F(I,J) MATRIX - UNITS OF HARTREES/BOHR**2

DO 30 I=1,MM
   DO 30 J=1,N
      SUM=0.
      DO 30 L=1,N
         SUM=SUM+PROD(I,L)*BPLMI(L,J)
      END
      PROD(I,J)=SUM
   END
   CONTINUE

TO PUT F(I,J) IN UNITS OF MDYNE/A

INSERT THE FOLLOWING STATEMENT

F(I,J)=15.57*F(I,J)

CONTINUE
C... F(I,J) MATRIX
C...

OPEN(1,FILE='FORCE.OUT')
OPEN(31,FILE='FORCE2.OUT')
WRITE(1,*) ' FORCE CONSTANT MATRIX'
WRITE(1,*) ' (INTERNAL COORDINATES - UNITS OF HARTREES/BOHR**2)'
WRITE(1,*)
II=1
DO 36 I=1,NROW
   DO 36 J=1,1
      FID(II)=F(I,J)
      II=II+1
36 CONTINUE
CALL OUTPAK(F1D,NROW,NMATR,1,1)
C DO 40 I=1,MM
C WRITE(1,111) (F(I,J),J=1,MM)
C 40 CONTINUE
WRITE(1,102)
C DO 41 I=1,15
C WRITE(1,112) (F(I,J),J=13,15)
C 41 CONTINUE
C... CONVERT ELEMENTS OF FORCE CONSTANT MATRIX
C... TO UNITS OF MDYNES/ANGSTROM
C...

DO 42 I=1,MM
   DO 42 J=1,1
      F(I,J)=15.56923*F(I,J)
   42 WRITE(31,120) I,J,F(I,J)
42 CONTINUE
C...
C...
FORMATS
C...
102 FORMAT(1X)
111 FORMAT(12F12.6)
112 FORMAT(9F12.6)
115 FORMAT(15F12.6)
116 FORMAT(15E15.6)
120 FORMAT(2I4,G20.12)
STOP
END

SUBROUTINE BKMATR(M,N,L,K)
IMPLICIT REAL*8 (A-H, O-Z)
REAL*8 K
DIMENSION B(M,N),K(N,N);
NAT=N/3
MM=3*NAT-6
C...
B MATRIX
OPEN(21,FILE='BMAT.IN')
KK=-11
160 KK=KK+12
   L=MIN0(KK+11,N)
   DO 170 I=1,MM
      READ(21,114) (B(I,J),J=KK,L)
      IF(L.LT.N)GO TO 160
   170 C DO 30 I=1,MM
   C READ(21,114) (B(I,J),J=1,12)
   C 30 CONTINUE
C  DO 31 I=1,MM
C  READ(21,114) (B(I,J),J=13,N)
C 31    CONTINUE
C      OPEN(22,FILE='BMAT.OUT')
K K=-11  
161  KK=KK+12
L=MINO(KK+11,N)
DO 171 I=1,MM
171  WRITE(22,114) (B(I,J),J=KK,L)
IF(L.LT.N)GO TO 161
C      DO 40 I=1,MM
C 40     WRITE(22,110) (B(I,J),J=1,12)
C      CONTINUE
C      WRITE(22,102)
C      DO 41 I=1,MM
C 41     WRITE(22,110) (B(I,J),J=13,N)
C      CONTINUE
C...K MATRIX
C...
OPEN(11,FILE='KMAT.IN')
DO 50 I=1,NAT
READ(11,121) (K(J,I*3-2),K(J,I*3-1),K(J,I*3),J=1,N)
50    CONTINUE
C      READ(11,105)
C      DO 51 I=1,21
C 51    READ(11,101) (K(I,J),J=10,18)
C      CONTINUE
C      READ(11,105)
C      DO 52 I=1,21
C 52    READ(11,104) (K(I,J),J=19,21)
C      CONTINUE
C      OPEN(12,FILE='KMAT.OUT')
K K=-11  
260  KK=KK+12
L=MINO(KK+11,N)
DO 270 I=1,N
270  WRITE(12,110) (K(I,J),J=KK,L)
IF(L.LT.N)GO TO 260
C      DO 60 I=1,12
C 60     WRITE(12,101) (K(I,J),J=1,9)
C      CONTINUE
C      WRITE(12,102)
C      DO 61 I=1,12
C 61    WRITE(12,101) (K(I,J),J=10,12)
C      CONTINUE
C      WRITE(12,102)
C      DO 62 I=1,21
C 62    WRITE(12,104) (K(I,J),J=19,21)
C      CONTINUE
C...FORMATS
C...
101    FORMAT(9F12.8)
C102    FORMAT(1H )
102    FORMAT(1X)
103    FORMAT(A5)
104    FORMAT(3F12.8)
105    FORMAT(/)
110    FORMAT(12E15.6)
SUBROUTINE MATCALC(A, B, N, M)
C...
C... THIS SUBROUTINE WILL DETERMINE
C... (1) DET OF A
C... (2) INVERSE OF A
C... (3) SOLVE A SYSTEM OF EQUATIONS
C... BASED ON THE VALUE OF THE PARAMETER INDEX
C... IF INDEX EQUALS (0,1,-1) THE OPTION SHOWN ABOVE WILL BE DETERMINED
C... A(N,M) = THE AUGMENTED MATRIX
C... B(N,N) = ORIGINALLY THE NxN IDENTITY... THE INVERSE MATRIX FINALLY
C... THE METHOD USED IS GAUSSIAN ELIMINATION WITH PIVOTING
C...
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(N,M),B(N,M)
COMMON NSYS, INDEX, DET
SIGN=1
MARK =0
N4I=N-1
NN=2*N
NPLSY=N+NSYS
IF(INDEX.LE.0) GO TO 2
DO 1 I=1,N
   DO 1 J=1,N
      1 A(IUN+J) =B(I, J)
NPLSY=NN
2 CONTINUE
DO 10 I=1,1mI
C... FROM HERE TO STATEMENT 4 THE PROGRAM PICKS UP THE PIVOT
C...
MAX=I
AMAX=ABS(A(I, I))
K=I
3 K=K+1
   IF(ABS(A(K, I)).LE.AMAX) GO TO 4
   MAX=K
   AMAX=ABS(A(K, I))
4 IF(K.NE.N) GO TO 3
   IF(MAX.EQ.I) GO TO 6
C... THE NEXT SEQUENCE INTERCHANGES ROWS
C...
   L=I-1
   L=L+1
   TEMP=A(I, L)
   A(I, L)=A(MAX, L)
   A(MAX, L)=TEMP
   IF(L.LT.NPLSY) GO TO 5
   SIGN=-SIGN
6 J=I
7 J=J+1
   IF(A(J, I).EQ.0.0) GO TO 9
   CONST=-A(J, I)/A(I, I)
L=I-1
8 L=L+1
  A(J,L)=A(J,L)+A(I,L)*CONST
  IF(L.NE.NPLSY) GO TO 8
9 CONTINUE
  IF(J.NE.N) GO TO 7
10 CONTINUE
  TEMP=1
  DO 11 I=1,N
    IF(A(I,I).EQ.0.0) GO TO 12
11    TEMP=TEMP*A(I,I)
    DET=SIGN*TEMP
    GO TO 13
12    MARK=1
    DET=0.0
13    IF(INDEX.EQ.0) GO TO 21
    IF(MARK.NE.1) GO TO 15
    WRITE(6,14)
14    FORMAT(///2X,21HMATRIX A IS SINGULAR.)
    GO TO 21
15    N1=N+1
16    HERE THE PROGRAM CARRIES OUT BACK SUBSTITUTION
17    DO 20 I=N1,NPLSY
      K=N
18      B(K,I)=A(K,I)
      IF(K.EQ.N) GO TO 18
      J=K
19      J=J+1
      B(K,I)=B(K,I)-A(K,J)*B(J,I)
      IF(J.NE.N) GO TO 17
20      B(K,I)=B(K,I)/A(K,K)
      IF(K.EQ.1) GO TO 19
      K=K-1
      GO TO 16
21    CONTINUE
   20     DO 20 L=I,N
22     A(L,I)=B(L,I)
23   21    RETURN
24    END

SUBROUTINE OUTPAK (MATRIX, NROW, NMATR, NCTL, NOUT)
C
C OUTPAK PRINTS A REAL*8 SYMMETRIC MATRIX STORED IN ROW-PACKED LOWER
C TRIANGULAR FORM (SEE DIAGRAM BELOW) IN FORMATTED FORM WITH NUMBERED
C ROWS AND COLUMNS. THE INPUT IS AS FOLLOWC:
C
C MATRIX(*) PACKED MATRIX
C
C NROW NUMBER OF ROWS TO BE OUTPUT
C
C NCTL CARRIAGE CONTROL FLAG: 1 FOR SINGLE SPACE,
C 2 FOR DOUBLE SPACE,
  3 FOR TRIPLE SPACE.

C NOUT ..................UNIT NUMBER FOR OUTPUT

C THE MATRIX ELEMENTS ARE ARRANGED IN STORAGE AS FOLLOWS:
C
C  1  2  3
C  4  5  6
C  7  8  9 10
C 11 12 13 14 15
C 16 17 18 19 20 21
C 22 23 24 25 26 27 28
C AND SO ON.

C OUTPAK IS SET UP TO HANDLE 6 COLUMNS/PAGE WITH A 6F20.14 FORMAT
C FOR THE COLUMNS. IF A DIFFERENT NUMBER OF COLUMNS IS REQUIRED, CHANGE
C FORMATS 1000 AND 2000, AND INITIALIZE KCOL WITH THE NEW NUMBER OF
C COLUMNS.

C AUTHOR: NELSON H.F. BEEBE, QUANTUM THEORY PROJECT, UNIVERSITY OF
C FLORIDA, GAINESVILLE

C REAL*8 MATRIX, COLUMN
INTEGER BEGIN, ASA, BLANK, CTL
DIMENSION MATRIX(NMATR), ASA(3)
DATA KCOL/5/, COLUMN/8HCOLUMN/
& BLANK/4H , 4H0 , 4H-
& ZERO/O.D+00/
CTL = BLANK
IF ((NCTL.LE.3).AND.(NCTL.GT.0)) CTL = ASA(NCTL)

C LAST IS THE LAST COLUMN NUMBER IN THE ROW CURRENTLY BEING PRINTED
C
C LAST = MIN0(NROW,KCOL)
C
C BEGIN IS THE FIRST COLUMN NUMBER IN THE ROW CURRENTLY BEING PRINTED.
C
C NCOL=1
C......BEGIN NON STANDARD DO LOOP.
BEGIN=1
1050 NCOL = 1
   WRITE (NOUT,1000) (I,I = BEGIN,LAST)
   DO 40 K = BEGIN,NROW
      KTOTAL = (K*(K-1))/2 + BEGIN - 1
   C DO 10 I = 1,NCOL
   C GO TO 20
   C IF (MATRIX(KTOTAL+I) .NE. ZERO) GO TO 20
   10 CONTINUE
   20 WRITE (NOUT,2000) CTL,K, (MATRIX(I+KTOTAL),I=I,NCOL)
   C GO TO 30
   C IF (K .LT. (BEGIN+KCOL-1)) NCOL = NCOL + 1
   30 IF (K .LT. (BEGIN+KCOL-1)) NCOL = NCOL + 1
   40 CONTINUE
   C LAST = MIN0(LAST+KCOL,NROW)
   BEGIN=BEGIN+NCOL
IF (BEGIN.LE.NROW) GO TO 1050
1000 FORMAT (/12X,4(11X,I4,5X), (11X,I4))
2000 FORMAT (A1, 4X, I4, 2X, 5D20.12)
RETURN
END
II.2.3 Program matmult2.f

The third program is matmult2.f. This program also carries out the transformation

\[ F = B^{-2} K B^{-1} \quad (8) \]

and also determines \( K \) from \( F \). The program allows input of the scaling factors, the \( Q_i \)'s, to scale the force constant matrix in internal coordinates, \( F \), and converts the scaled \( F \) to a scaled force constant matrix in Cartesian coordinates, \( K \). This scaled \( K \) is used as input to the CADPAC program to carry out a VCD calculation of allowed frequencies of vibration and rotational strengths. In addition, the parameter NAT must be changed for each molecule considered.

On the following pages a listing of the FORTRAN program matmult2.f is given.
PROGRAM FCMATRIX
C...
C... PUNCHES F.C.M TO FORTRAN UNIT 7
C...
PARAMETER (NAT=16, MM=3*NAT-6, N=3*NAT, MM=2*M, NAT3=N, NDM=NAT3)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 K(N,N),M(N,N),KNEW(N,N)
REAL MC,MO,MH
DIMENSION B(MM,N), BM(MM, N), BP(N, MM), P(N, N), TEST(MM, MM),
1 BMBP(MM, MM), BMBPI(MM, MM), BPLM(MM, N), PROD(MM, N), BPLMI(N, MM),
2 TESTA(MM, MM), AA(MM, MM), BB(MM, MM), FNEW(MM, MM),
3 TITLE(10), C(3, NAT), GRAD(3, NAT), FCM(NDIM, NDIM), PROD2(MM, N),
4 Q(15)
COMMON NSYS, INDEX, DET

C...
C... READ(8,105) (TITLE(I),I=1,9)
C... WRITE(7,105) (TITLE(I),I=1,9)
C...
C... READ(8,106)
C... WRITE(7,106)
C...
C... READ(8,107) (C(1,I),C(2,I),C(3,I),I=1,NAT)
C... WRITE(7,107) (C(1,I),C(2,I),C(3,I),I=1,NAT)
C...
C... READ(8,108)
C... WRITE(7,108)
C...
C... READ(8,107) (GRAD(1,I),GRAD(2,I),GRAD(3,I),I=1,NAT)
C... WRITE(7,107) (GRAD(1,I),GRAD(2,I),GRAD(3,I),I=1,NAT)
C...
C... READ(8,109)
C... WRITE(7,109)
C...
C... GET MATRICES B AND K
C...
CALL BKMATR(MM, N, B, K)
C...
ADJOINT OF B
C...
DO 10 I=1,N
DO 10 J=1,MM
BP(I,J)=B(J,I)
10 CONTINUE
C...
C... DETERMINE PRODUCT OF B M BP MATRICES
C...
MC=12.01
MO=16.00
MH=1.008
DO 501 I=1,N
DO 501 J=1,N
M(I,J)=0.
501 CONTINUE
DO 502 I=1,N
M(I,I)=1.
502 CONTINUE
C...
DO 11 I=1,MM
DO 11 J=1,N
SUM=0.
DO 11 L=1,N
SUM=SUM+B(I,L)*M(L,J)
BM(I,J)=SUM
11 CONTINUE

C... DETERMINE PRODUCT OF BM BP MATRICES
DO 511 I=1,MM
DO 511 J=1,MM
SUM=0.
DO 511 L=1,N
SUM=SUM+BM(I,L)*BP(L,J)
BMBP(I,J)=SUM
511 CONTINUE

OPEN(3,FILE='TESTADZ.OUT')
OPEN(23,FILE='BMBP.MAT')
WRITE(3,*) '3MBPs
DO 540 I=1,MM
WRITE(3,115) (BMBP(I,J),J=1,M4)
WRITE(23,115) (BMBP(I,J),J=1,M4)
540 CONTINUE

C... DETERMINE INVERSE OF BM BP
DO 12 I=1,MM
DO 12 J=1,MM
AA(I,J)=BMBP(I,J)
12 CONTINUE

NSYS=0
INDEX=1
DO 221 I=1,MM
DO 221 J=MM+1,MM+I
AA(I,J)=0.
221 CONTINUE

DO 222 I=1,MM
DO 222 J=1,MM
BB(I,J)=0.
IF(I.EQ.J) BB(I,J)=1.
222 CONTINUE

CALL MATCALC(AA, BB, MM, MM)
WRITE(6,*) 'DETA =', DET

C... SET BMBPI MATRIX
C...
DO 191 I=1,MM
DO 191 J=1,MM
BMBPI(I,J)=AA(I,J+MM)
191 CONTINUE

WRITE(3,*) 'BMBPI'
DO 192 I=1,MM
WRITE(3,116) (BMBPI(I,J),J=1,MM)
192 CONTINUE

C... DETERMINE TESTA MATRIX
C...
DO 302 I=1,MM
DO 302 J=1,MM
SUM=0.
DO 302 I=1, MM
   SUM=SUM+BMBPI(I,L)*BMBP(L,J)
   TESTA(I,J)=SUM
302    CONTINUE
   WRITE(3,'(A)') 'TESTA'
   DO 340 I=1, MM
      WRITE(3,111) (TESTA(I,J), J=1, MM)
340    CONTINUE
C    WRITE(3,102)
C    DO 341 I=1, 15
C    WRITE(3,112) (TESTA(I,J), J=13, 15)
C    341    CONTINUE
C...
   DETERMINE BPLM MATRIX
C...
   DO 13 I=1, MM
      DO 13 J=1, N
         SUM=0.
      DO 13 L=1, MM
         SUM=SUM+BMBPI(I,L)*BM(L,J)
         BPLM(I,J)=SUM
13     CONTINUE
C...
   DETERMINE TEST MATRIX
C...
   DO 202 I=1, MM
      DO 202 J=1, MM
         SUM=0.
      DO 202 L=1, N
         SUM=SUM+BPLM(I,L)*BP(L,J)
         TESTC(I,J)=SUM
202    CONTINUE
   OPEN(2,FILE='TESTDZ.OUT')
   DO 240 I=1, MM
      WRITE(2,111) (TEST(I,J), J=1, MM)
240    CONTINUE
C    WRITE(2,102)
C    DO 241 I=1, 15
C    WRITE(2,112) (TEST(I,J), J=13, 15)
C    241    CONTINUE
C...
   DETERMINE TRANSPOSE OF BPLM MATRIX
C...
   DO 14 I=1, N
      DO 14 J=1, MM
         BPLM(J,I)=BPLM(I,J)
14     CONTINUE
C...
   DETERMINE PRODUCT OF BPLM K MATRICES
C...
   DO 20 I=1, MM
      DO 20 J=1, N
         SUM=0.
      DO 20 L=1, N
         SUM=SUM+BPLM(I,L)*K(L,J)
         PROD(I,J)=SUM
20     CONTINUE
C...
C...
   DETERMINE PRODUCT OF BPLM K BPLMI MATRICES TO GIVE
C...
   F(I,J) MATRIX
-27-
C...
DO 30 I=1,MM
DO 30 J=1,MM
    SUM=0.
DO 30 L=1,N
    SUM=SUM+PROD(I,L)*BPLMI(L,J)
    F(I,J)=SUM
C...
IN ORDER TO CONVERT TO UNITS OF MDYNE/A
C...
INSERT THE FOLLOWING STATEMENT
C...
C F(I,J)=15.57*F(I,J)
30 CONTINUE
C...
F(I,J) MATRIX
C...
OPEN(1,FILE='FORCE.OUT')
WRITE(1,*) ' FORCE CONSTANT MATRIX'
WRITE(1,*) '(INTERNAL COORDINATES - UNITS OF HARTREES/A)'
WRITE(1,*)
DO 40 I=1,MM
    WRITE(1,111) (F(I,J),J=1,NK)
40 CONTINUE
C WRITE(1,102)
C DO 41 I=1,15
C    WRITE(1,112) (F(I,J),J=13,15)
C 41 CONTINUE
C...
SCALING FACTORS Q(I) INPUT HERE
C...
Q(1)= 0.958
Q(2)= 0.958
Q(3)= 0.958
Q(4)= 0.907
Q(5)= 0.772
Q(6)= 0.863
Q(7)= 0.931
Q(8)= 0.845
Q(9)= 0.907
Q(10)= 0.863
Q(11)= 0.845
Q(12)= 0.907
Q(13)= 0.863
Q(14)= 0.863
Q(15)= 0.845
Q(16)= 0.923
Q(17)= 0.923
Q(18)= 0.914
Q(19)= 0.904
Q(20)= 0.901
Q(21)= 0.903
Q(22)= 0.946
Q(23)= 0.902
Q(24)= 0.901
Q(25)= 0.946
Q(26)= 0.902
Q(27)= 0.902
Q(28)= 0.901
Q(29)= 0.946
Q(30)= 0.900
Q(31) = 0.932
Q(32) = 0.916
Q(33) = 0.900
Q(34) = 0.900
Q(35) = 1.093
Q(36) = 0.921
Q(37) = 0.910
Q(38) = 1.093
Q(39) = 0.921
Q(40) = 0.921
Q(41) = 0.910
Q(42) = 0.984

C...
C... NEW F MATRIX, FNEW
C...

OPEN(51, FILE='FNEW.OUT')
WRITE(51, *) 'FNEW IN UNITS OF HARTREES/BOHR'
DO 601 I=1, MM
   DO 601 J=1, MM
      FNEW(I, J) = SQRT(Q(I) * Q(J)) * F(I, J)
      IF (I.EQ.J) FNEW(I, J) = Q(I) * F(I, J)
   CONTINUE
1230 FORMAT(15E15.6)
1231 FORMAT(15E15.6)
DO 807 I=1, MM
   WRITE(51, 1230) (FNEW(I, J), J=1, I)
807 CONTINUE
WRITE(51, 1231) I, Q(I)

C...
C... NEW K MATRIX, KNEW
C...

DO 602 I=1, MM
   DO 602 J=1, N
      SUM = 0.
      DO 602 L=1, MM
         SUM = SUM + FNEW(I, L) * B(L, J)
      C
      SUM = SUM + FNEW(I, L) / 15.57 * B(L, J)
      PROD2(I, J) = SUM
   CONTINUE
DO 603 I=1, N
   DO 603 J=1, N
      SUM = 0.
      DO 603 L=1, MM
         SUM = SUM + BP(I, L) * PROD2(L, J)
      KNEW(I, J) = SUM
      FCM(I, J) = KNEW(I, J)
   CONTINUE
DO 660 I=1, NAT
   READ(8, 107) (FCM(J, I*3-2), FCM(J, I*3-1), FCM(J, I*3), J=1, NAT3)
   OPEN(71, FILE='KMATNEW.OUT')
   WRITE(71, 107) (FCM(J, I*3-2), FCM(J, I*3-1), FCM(J, I*3), J=1, NAT3)
660 CONTINUE
C... FORMATS
SUBROUTINE BKHATR(M,N, B, K)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 K
DIMENSION B(M,N),K(N,N)
NAT=N/3
MM=N-6
WRITE(*,*), 'NAT=', NAT
WRITE(*,*) 'MM=', MM
C B MATRIX
OPEN(21,FILE='BMAT.IN')
KK=-11
160 KK=KK+12
L=MIN0 (KK+11,N)
DO 170 I=1,MM
170 READ(21,114) (B(I,J), J=KK,L)
IF(L.LT.N)GO TO 160
C DO 30 I=1,12
C READ(21,114) (B(I,J), J=1,12)
C 30 CONTINUE
C DO 31 I=1,15
C READ(21,114) (B(I,J), J=13,21)
C 31 CONTINUE
OPEN(22,FILE='BMAT.OUT')
KK=-11
161 KK=KK+12
L=MIN0 (KK+11,N)
DO 171 I=1,MM
171 WRITE(22,114) (B(I,J), J=KK,L)
IF(L.LT.N)GO TO 161
C DO 40 I=1,12
C WRITE(22,111) (B(I,J), J=1,12)
C 40 CONTINUE
C WRITE(22,102)
C 41 CONTINUE
C K MATRIX
C
OPEN(11,FILE='KMAT.IN')
DO 50 I=1,NAT
READ(11,121) (K(J,I*3-2), K(J,I*3-1), K(J,I*3), J=1,N)
50 CONTINUE
C READ(11,105)
C DO 51 I=1,21
C  READ(11,101) (K(I,J),J=10,18)
C51  CONTINUE
C  READ(11,105)
C  DO 52 I=1,21
C  READ(11,104) (K(I,J),J=19,21)
C52  CONTINUE
OPEN(12,FILE='IOMAT.OUT')
DO 60 I=1,N
WRITE(12,101) (K(I,J),J=1,9)
60  CONTINUE
WRITE(12, 102)
DO 61 I=1,N
WRITE(12,101) (K(I,J),J=10,12)
61  CONTINUE
C  WRITE(12,102)
C  DO 62 I=1,21
C  WRITE(12,104) (K(I,J),J=19,21)
C 62  CONTINUE
C...
C...
FORMATS
C...
101  FORMAT(9F12.8)
C102  FORMAT(1H1)
102  FORMAT(IX)
103  FORMAT(A5)
104  FORMAT(3F12.8)
105  FORMAT(/)
111  FORMAT(12F10.6)
112  FORMAT(9F10.6)
114  FORMAT(12E15.6)
121  FORMAT(1X,3E20.12)
RETURN
END

SUBROUTINE MATCALC(A, B, N, M)
C...
C...  THIS SUBROUTINE WILL DETERMINE
C...
(1) DET OF A
C...
(2) INVERSE OF A
C...
(3) SOLVE A SYSTEM OF EQUATIONS
C...
BASED ON THE VALUE OF THE PARAMETER INDEX
C...
IF INDEX EQUALS (0,1,-1) THE OPTION SHOWN ABOVE WILL BE DETERMINED
C...
A(N,M) = THE AUGMENTED MATRIX
C...
B(N,N) = ORIGINALLY THE N×N IDENTITY...THE INVERSE MATRIX FINALLY
C...
THE METHOD USED IS GAUSSIAN ELIMINATION WITH PIVOTING
C...
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(N,M),B(N,M)
COMMON NSYS, INDEX, DET
SIGN=1
MARK =0
NMI=N-1
NN=2*N
NPLSY=N+NSYS
IF(INDEX.LE.0) GO TO 2
DO 1 I=1,N
DO 1 J=1,N
1  A(I,N+J)=B(I,J)
NPLSY=NN
2  CONTINUE
DO 10 I=1,N
C...
C... FROM HERE TO STATEMENT 4 THE PROGRAM PICKS UP THE PIVOT
C...
MAX=I
AMAX=ABS(A(I,I))
K=I
3 K=K+1
IF(ABS(A(K,I)).LE.AMAMAX) GO TO 4
MAX=K
AMAX=ABS(A(K,I))
4 IF(K.NE.N) GO TO 3
IF(MAX.EQ.I) GO TO 6
C...
C... THE NEXT SEQUENCE INTERCHANGES ROWS
C...
L=I-1
5 L=L+1
TEMP=A(I,L)
A(I,L)=A(MAX,L)
A(MAX,L)=TEMP
IF(L.LT.NPLSY) GO TO 5
SIGN=-SIGN
6 J=I
7 J=J+1
IF(A(J,I).EQ.0.0) GO TO 9
CONST=-A(J,I)/A(I,I)
L=I-1
8 L=L+1
A(J,L)=A(J,L)+A(I,L)*CONST
IF(L.NE.NPLSY) GO TO 8
9 CONTINUE
IF(J.NE.N) GO TO 7
10 CONTINUE
TEMP=1
DO 11 I=1,N

IF(A(I,I).EQ.0.0) GO TO 12
11 TEMP=TEMP*A(I,I)
DET=SIGN*TEMP
GO TO 13
12 MARK=1
DET=0.0
13 IF(INDEX.EQ.0) GO TO 21
IF(MARK.NE.1) GO TO 15
WRITE(6,14)
C...
C... FORMATS
C...
14 FORMAT(///2X,21H MATRIX A IS SINGULAR.)
GO TO 21
15 N1=N+1
C...
C... HERE THE PROGRAM CARRIES OUT BACK SUBSTITUTION
C...
DO 20 I=N1,NPLSY
K=N
16 B(K,I)=A(K,I)
IF(K.EQ.N) GO TO 18
J=K
17 J=J+1
-32-
B(K, I) = B(K, I) - A(K, J) * B(J, I)
IF(J .NE. N) GO TO 17

18  B(K, I) = B(K, I) / A(K, K)
IF(K .EQ. 1) GO TO 19
K = K - 1
GO TO 16

19  CONTINUE
DO 20 L = 1, N
20    A(L, I) = B(L, I)
21  RETURN
END
II.2.4 Program simplex.f

Another program named simplex.f was generated from one of the programs of the programs of McIntosh and Peterson [1]. This program allows the scaling factors to be determined by a best fit to a set of inputted experimental frequencies. This approach was investigated; however, the approach used in this study was to determine a Q, by comparison of the diagonal force constants at the 6-31G* level $F_{ii}$(HF) and $F_{ii}$(MP2). The simplex approach may be the better approach and should be given serious consideration for scaling procedures to be studied in the future.
REFERENCES

Table 1. Data file for R-glyceraldehyde which is used as input to the FORTRAN program bmat.f. The program bmat.f determines the transformation matrix B defined by \( R = B q \) where \( R \) is a column vector of the Cartesian coordinates.

<table>
<thead>
<tr>
<th>C3H6O3 - glyceraldehyde [hf/6-31g*]</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 0</td>
</tr>
<tr>
<td>-2.026663  -1.245592  0.698964  C-1</td>
</tr>
<tr>
<td>-0.075542  0.791716   1.181527  C-2</td>
</tr>
<tr>
<td>2.590801   -0.265346  0.987662  C-3</td>
</tr>
<tr>
<td>-1.840047   -2.980089  1.800025  H-4</td>
</tr>
<tr>
<td>-3.709799   -0.957621  -0.760258  O-5</td>
</tr>
<tr>
<td>-0.389204   1.489681   3.097911  H-6</td>
</tr>
<tr>
<td>-0.333856   2.731754  -0.577374  O-7</td>
</tr>
<tr>
<td>-1.950507   2.587079  -1.351823  H-8</td>
</tr>
<tr>
<td>3.928350    1.233993   1.405427  H-9</td>
</tr>
<tr>
<td>2.869625   -1.763088   2.355118  H-10</td>
</tr>
<tr>
<td>3.009882   -1.303036  -1.399301  O-11</td>
</tr>
<tr>
<td>2.720395   -0.021029  -2.620120  H-12</td>
</tr>
</tbody>
</table>

1 1 2 0 0 0 0 1-2 bond stretch
1 2 3 0 0 0 0 2-3 bond stretch
1 1 4 0 0 0 0 1-4 bond stretch
1 1 5 0 0 0 0 1-5 bond stretch
1 2 6 0 0 0 0 2-6 bond stretch
1 2 7 0 0 0 0 2-7 bond stretch
1 7 8 0 0 0 0 7-8 bond stretch
1 3 9 0 0 0 0 3-9 bond stretch
1 3 10 0 0 0 0 3-10 bond stretch
1 3 11 0 0 0 0 3-11 bond stretch
1 11 12 0 0 0 0 11-12 bond stretch
2 3 2 1 0 0 0  bond angle bend 3-2-1
2 4 1 2 0 0 0  bond angle bend 4-1-2
2 5 1 2 0 0 0  bond angle bend 5-1-2
2 6 2 1 0 0 0  bond angle bend 6-2-1
2 7 2 1 0 0 0  bond angle bend 7-2-1
2 8 7 2 0 0 0  bond angle bend 8-7-2
2 9 3 2 0 0 0  bond angle bend 9-3-2
2 10 3 2 0 0 0  bond angle bend 10-3-2
2 11 3 2 0 0 0  bond angle bend 11-3-2
2 12 11 3 0 0 0  bond angle bend 12-11-3
Table 1. Data file for R-glyceraldehyde which is used as input to the FORTRAN program bmat.f. The program bmat.f determines the transformation matrix $B$ defined by $R = B \mathbf{q}$ where $R$ is a column vector of the Cartesian coordinates. 

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 4-1-2-3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 5-1-2-3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 6-2-1-4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 7-2-1-4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 8-7-2-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 9-3-2-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 10-3-2-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 11-3-2-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>11</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>dihedral angle 12-11-3-2</td>
</tr>
</tbody>
</table>