# CHAPTER 6

## CONCLUSIONS AND FUTURE DIRECTIONS

Despite this analysis of dipolar and chemical shift coherence transfer, there still remains much work to be done, especially on multiply arranged nuclei in uniformly labeled proteins. The lack of current direction in this regard is disturbing, given that there most certainly exist complex systems whose geometry would lend itself to possible simplification of the coherence transfer mathematics. A protein would never be so polite as to arrange its nuclei in a symmetric fashion for an experimenter, but there may be clusters of homonuclei that can be modeled as distorted geometric arrangements.

In Chapters 2 and 3, we begin to see the attraction in studying coherent transfer of magnetization. Although the two-spin case is simplest, there is promise for further study of "relay" spectroscopy, to send coherent magnetization skipping from one site to another. In fact, Spiess & Rohr [130] have numerically modeled the transfer of magnetization along a model chain like a polypeptide for many spins.

Chapter 4 dealt with many different experiments, examining the form of DCT across several different experiments. The interesting fact that DRAMA has, within its own experimental parameters, the ability to selectively transfer magnetization could be seen by the analysis of the analytical equations after application of the DRAMA Hamiltonian. SSNMR experimentation would certainly benefit from further development of these existing pulse sequences.

In Chapter 5, we have seen that perturbations to the symmetry of the dipolar Hamiltonian, by the introduction of the chemical shift, decreased the coherence transfer by introducing a non-symmetric component to the coherent evolution of magnetization and effectively causing the coherence to dissipate away into transverse coherences, removing the ability for magnetization to dwell in non-observable resevoirs.

We have seen in Chapter 4 that the coherence transfer does have a “lifetime”, in which the maximum coherence transfer in unoriented samples has a limiting velocity, which may well limit the number density for multiply-labeled sites within a protein. That is to say, it may be possible to model a uniformly labeled protein by treating the “spheres” of decreasing coherence transfer as interaction “unit cells” within a particular time, so that short-duration experiments, with limited mixing times, can still glean structural information from uniformly labeled proteins when the only interaction present is the dipolar and/or scalar coupling.

The next steps in this study will generally go in this direction:

- Determine the absolute boundary for detectable coherence transfer (before dephasing loses information) against the predicted model.
- Develop the three-spin model to simulate an unoriented system of three strongly coupled nuclei.
- Simulate groups of nuclei within and without those detectable coherence transfer boundaries using a three-spin model.
- Set up geometrical models and determine the coherence transfer using numerical calculations.

Further work is planned in these areas.

.

# APPENDICES

## APPENDIX A: EXAMPLE PROGRAMS

```
                      PROGRAM CRYSTAL



C     This program calculates the response of a two-spin
C     system to the dipolar Hamiltonian (Hd) when one of
C     the spins (the I spin) is perturbed. It can also
C     include isotropic J as well as D. It then graphs
C     the FID response of spin S as well as spin I to two
C     seperate files so you can plot them. This also
C     involves itself with FFT calculations. Comments provided
C     for more clue.  This program was verified with an
C     H-H calculation on Sz at 2.5 angstroms against
C     R. Bruschweiler's  review article, page 5, in
C     Progress in Nuclear Mag Resonance Spec, 32 (1998)
C     - Deanne Taylor, Sept, 1998.

      Implicit double precision (a-h, o-z)
      INTEGER *2 FL
      REAL TSTEP
      DOUBLE PRECISION PI, HBAR, APAS
      CHARACTER *30 FID, FFTOUT
      CHARACTER *1 STATE, SIGNI, FFTYN
      DIMENSION  RES1(100000),  RESULT1(100000), CONFIG(100000)
      DIMENSION AT(131138), BT(131138), CT(131138), DT(131138)
      DATA PI /3.1415 92653 58979 324/
C     Planck's constant in cgs (erg sec rad(-1))
      DATA HBAR /1.0545887E-27/

      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
5     PRINT *, 'Calculations are in cgs.'
      PRINT *, ''
      PRINT *,''

      PRINT *, 'Input the eqn you want to calc (A,B,C,D,E,F,G).'
      READ (*,'(A)') STATE

      IF ((STATE.EQ.'A').OR.(STATE.EQ.'B').OR.(STATE.EQ.'C').OR.
     &(STATE.EQ.'D').OR.(STATE.EQ.'E').OR.(STATE.EQ.'F').OR.
     &(STATE.EQ.'G')) THEN
      STATE=STATE
      ELSE
      PRINT *, 'You must enter a state of A, B, C, D, E, F or G.
     &Please try again.'
        GOTO 5
        ENDIF

      PRINT *, 'Is this equation a negative? -A, -B, etc? (Y or
N)'
      READ (*,'(A)') SIGNI

      IF ((SIGNI.EQ.'Y').OR.(SIGNI.EQ.'y')) THEN
      SIGN= -1.0
      ELSE
         SIGN=1.0
      ENDIF

       PRINT *, 'What is the name for the data file?'
       READ (*, '(A)') FID
       OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')

C  Parameter Input

       PRINT *, 'Parameters'
       PRINT *, 'Value of Gamma for homonuclear pair?'
       PRINT *, '(C13 = 6727.0, H = 26750.0, in rad sec(-1) G(-
1))'
       READ  *, GAMMA
       PRINT *, 'Value of J for this pair (in Hz)?'
       READ *, JJJ
       PRINT *, 'Distance between nuclei? (in Angstroms)'
       READ *, R
       PRINT *, 'Do you want an FFT calculated?'
       READ *, FFTYN
       PRINT *, 'Milliseconds for simulation? (in ms)'
       READ *, TIMEX

       IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN
        PRINT *, 'Number of steps in time array? (pwr of 2 to
131136)'
        READ *, M
        PRINT *, 'Name of FFT output file?'
        READ *, FFTOUT
       ELSE
        PRINT *, 'Number of steps in time array? (Even # up to
10,000)'
        READ *, M

       ENDIF
```

```
        TIMER=TIMEX*.001
        TSTEP=TIMER/M


C  UNIT CONVERSIONS AND ETC.

C  R is the internuclear distance.

       R = R * 1.E-8

C  TNS is the constant in front of the dipolar hamiltonian...
C  gamma1*gamma2*hbar/R^3...since gamma1=gamma2, it's just
squared.
C  We divide by 2*Pi so we can transform it from rad/sec to Hz.
C (Note that TNS is negative, as D is negative).

       TNS=-(GAMMA**2)*HBAR*(1/R**3)

       PRINT *, 'Value of D =', TNS
       PRINT *, 'Distance in cm: ',R
       PRINT *, 'The gamma pair: ' , GAMMA


       PRINT *, 'Timesteps selected: ', TSTEP, ' seconds.'
       PRINT *, 'Your total data set will span ', TSTEP*M*1E03,
     * ' milliseconds.'

         PRINT *, 'At 89.'
89       PRINT *, 'Calculating FID for ', STATEX, '.'


C ---------------------- CALCULATIONS FOR EQN A -----------------

            IF (STATE.EQ.'A') THEN

C           Calculation for <eqn A>

         DO 11 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5 *
     &(COS(3.*TNS*DELTAT)+COS((TNS + JJJ*PI)*DELTAT))
           RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
11          CONTINUE


C ----------------- CALC FOR EQN B --------------------------

        ELSE IF (STATE.EQ.'B') THEN

C               Loop for <eqn B>

         DO 12 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5 *
     &(-COS(3.*TNS*DELTAT) + COS((TNS + JJJ*PI)*DELTAT))
           RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
12          CONTINUE



C ----------------- CALC FOR EQN -B -------------------------

          ELSE IF (STATE.EQ.'-B') THEN

C               Loop for <eqn -B>

         DO 13 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5*
     &(-COS(3.*TNS*DELTAT)+COS((TNS+JJJ*PI)*DELTAT))
           RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
13          CONTINUE



C--------------- CALC FOR EQN C --------------------

         ELSE IF (STATE.EQ.'C') THEN

C              Calculation for <eqn C>

         DO 14 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5 *
     &(1+COS((JJJ*PI-TNS)*DELTAT))
           RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
14       CONTINUE


C--------------- CALC FOR EQN D --------------------
```

```
         ELSE IF (STATE.EQ.'D') THEN


C               Calculation for <eqn D>

         DO 15 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5*(1-COS((JJJ*PI-TNS)*DELTAT))
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
15       CONTINUE



C--------------- CALC FOR EQN E ---------------------

         ELSE IF (STATE.EQ.'E') THEN

C               Calculation for <eqn E>

         DO 16 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5*
     &(SIN((TNS+JJJ*PI)*DELTAT)+SIN(3*TNS*DELTAT))
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
16       CONTINUE


C--------------- CALC FOR EQN -E  ---------------------

         ELSE IF (STATE.EQ.'-E') THEN

C               Calculation for <eqn -E>

         DO 17 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)= 0.5*
     &(-SIN((TNS+JJJ*PI)*DELTAT)-SIN(3.*TNS*DELTAT))
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
17       CONTINUE


C--------------- CALC FOR EQN F  ---------------------

         ELSE IF (STATE.EQ.'F') THEN

C               Calculation for <eqn F>

         DO 18 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)= 0.5*(SIN((TNS+JJJ*PI)*DELTAT)-
     &SIN(3.*TNS*DELTAT))
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
18       CONTINUE


C--------------- CALC FOR EQN -F  ---------------------

         ELSE IF (STATE.EQ.'-F') THEN

C               Calculation for <eqn -F>

         DO 19 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)= 0.5*
     &(-SIN((TNS+JJJ*PI)*DELTAT)+SIN(3.*TNS*DELTAT))
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
19       CONTINUE


C--------------- CALC FOR EQN G  ---------------------

         ELSE IF (STATE.EQ.'G') THEN

C               Calculation for <eqn G>

         DO 21 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.5*SIN((JJJ*PI-TNS)*DELTAT)
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
21       CONTINUE

        ENDIF

C Some useful write-tos to the output file.

       WRITE (4,*)' RESULTS FOR: ', STATE
       WRITE (4,*)' --------------------------'
       WRITE (4,*)' '
       WRITE (4,*)' Gamma for homonuclear pair: ', GAMMA
       WRITE (4,*)' No. of time points reporting:', M
       WRITE (4,*)' Value of J:' ,  JJJ , ' Hz'
       WRITE (4,*)' Internuclear Distance (angstroms): ', R*1E8
       WRITE (4,*)' Value of D: ', TNS
       WRITE (4,*)' ----------------'
```

```
      WRITE (4,*)' Evolution of EQN ', STATE , ' in time steps
of',
     &TSTEP*1E03, ' milliseconds'
      WRITE (4,*)' ----------------'

      PRINT *, 'Writing to FID file...'
      DO 111 L=0, M, 1
           WRITE(4,*)  L*1E03*TSTEP,  '   ' , RESULT1(L)
111     CONTINUE

         WRITE(4,*)'-9999'

9999   IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN

       DO 107 I=1, M, 1
       CONFIG(I)=RESULT1(I)
107    CONTINUE

       DO 109  I=1, M, 1
       WRITE (3,*) I*1E03*TSTEP, '    ', CONFIG(I)
109    CONTINUE


      CALL FFT(CONFIG,M,FFTOUT,TSTEP,STATE,GAMMA,
     &N,APAS,TAU1,TAU2,RATIO,GNUR)
      ENDIF
      END

     SUBROUTINE FFT (AT,M,FFTOUT,TSTEP,STATE,GAMMA,
     &N,APAS,TAU1,TAU2,RATIO,GNUR)
     REAL TSTEP
     DOUBLE PRECISION PI,HBAR,APAS,TAU1,TAU2,RATIO,GAMMA,GNUR
     CHARACTER *1 STATE, SIGNI, FFTYN
     INTEGER I, J
     DOUBLE PRECISION AT(M+2),BT(M/2+1),CT(M/2+1),DT(M/2+1)
     CHARACTER * 30 FFTOUT
     DATA PI /3.1415 92653 58979 324/
     OPEN(5, FILE=FFTOUT)
     CLOSE(4)
     CALL SFFT2(M/2,AT,1)
     CALL SFS(M,AT,1)
     PRINT *, 'Writing to FFT file...'
     WRITE (5,*), 'FFT CALCULATION FOR EQN ', STATE
      WRITE (5,*)' --------------------------'
      WRITE (5,*)' '
      WRITE (5,*)' No. of steps in integration: ', N
      WRITE (5,*)' Gamma for homonuclear pair: ', GAMMA
      WRITE (5,*)' No. of data points reporting:', M
      WRITE (5,*)' Internuclear Distance (angstroms): ', R*1E8
      WRITE (5,*)' RATIO inner evolution to rot period:', RATIO
      WRITE (5,*)' TAU1:' , TAU1, 'TAU2:' , TAU2
      WRITE (5,*)' Spinning Speed: ' , GNUR ,  ' Hz'
      WRITE (5,*)' Value of D: ', APAS
      WRITE (5,*)' ----------------'
      WRITE (5,*)' FFT of EQN ', STATE , ' in freq steps of',
     &1/(M*1E03*TSTEP), ' milliseconds'
      WRITE (5,*)' ----------------'

C Removing that first point

      DO 201 J=1, M-2
      DT(J) = AT(J+2)
201   CONTINUE
      DT(M-1)=0.
      DT(M)=0.

C CREATING THE DIPOLAR POWDER PATTERN: PRINT OUT THE LEFT SIDE BY
REVERSING
C THE SOLUTION FROM LEFT TO RIGHT, THEN PLOT THE FFT FOR THE
RIGHT HAND
C SIDE OF THE PLOT.

      DO 200 J=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(J)=DT(M-(2*J-1))
      WRITE(5,*), (J-(M/2+1))/(M*TSTEP), BT(J)*M
200   CONTINUE

      DO 202 I=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(I)=DT(2*I-1)
      CT(I)=DT(2*I)
      WRITE(5,*), I/(M*TSTEP), BT(I)*M
      WRITE(88,*),CT(I)
202   CONTINUE

C Finish off with a tag for the data file (for perl script
purposes)

      WRITE(5,*), '-9999'
```

```
      RETURN
      END


C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFFT2 (N,Z,SIGN)
      double precision Z(N)
      INTEGER N, SIGN
      INTEGER JBH, JBL, KR, LOG2
      INTEGER BLK, KK, K0, K1, K2, K3, N2, SPAN
      DOUBLE PRECISION CE, C1, C2, C3, SE, S1, S2, S3, PIOV2,
THETA
      DOUBLE PRECISION  X1, X2, X3, Y1, Y2, Y3
      DATA PIOV2/1.5707 96326 79489 662/
      N2 = 1
      LOG2 = 0
 8100    LOG2 = LOG2 + 1
         N2 = N2 + N2
         IF ( N2 .LT. N )  GO TO 8100
      IF ( N2 .GT. N ) RETURN
      N2 = N2 + N2 - 1
      IF ( SIGN .GE. 0 ) GO TO 8210
       DO 8200 KK = 1, N2, 2
 8200      Z(KK+1) = -Z(KK+1)
 8210 CONTINUE
      JBL = 2
      JBH = N
 8300 IF ( JBH .LE. JBL ) GO TO 8340
         SPAN = JBL + JBL
         KR = 1
 8310        KK = KR + JBL
             KR = KR + JBH
             BLK = KR - JBL
             DO 8330 K1 = KK, BLK, SPAN
                 K3 = K1 + JBL - 2
                 DO 8320 K0 = K1, K3, 2
                     X1 = Z(K0)
                     Z(K0) = Z(KR)
                     Z(KR) = X1
                     Y1 = Z(K0+1)
                     Z(K0+1) = Z(KR+1)
                     Z(KR+1) = Y1
 8320                KR = KR + 2
                KR = KR + JBL
 8330        CONTINUE
             IF ( KR .LT. N2 ) GO TO 8310
         JBH = JBH / 2
         JBL = JBL + JBL
         GO TO 8300
 8340 CONTINUE
      IF ( MOD(LOG2,2) .NE. 0 ) GO TO 8400
       THETA = PIOV2
       SPAN = 2
       GO TO 8420
 8400 CONTINUE
       DO 8410 K0 = 1, N2, 4
          K1 = K0 + 2
          X1 = Z(K0)
          Z(K0) = X1 + Z(K1)
          Z(K1) = X1 - Z(K1)
          Y1 = Z(K0+1)
          Z(K0+1) = Y1 + Z(K1+1)
 8410     Z(K1+1) = Y1 - Z(K1+1)
       THETA = PIOV2 + PIOV2
       SPAN = 4
 8420 CONTINUE
 8500 IF ( SPAN .GE. N2 ) GO TO 8560
         BLK = SPAN * 4
         IF ( SPAN .EQ. 2 ) GO TO 8505
           THETA = .25 * THETA
           SE = SIN(THETA)
           CE = COS(THETA)
           C1 = 1.0
           S1 = 0.0
 8505    CONTINUE
         DO 8550 KK = 1, SPAN, 2
             IF ( KK .EQ. 1 ) GO TO 8510
               X1 = C1 * CE - S1 * SE
               Y1 = C1 * SE + S1 * CE
               X2 = .5 * ( 3. - ( X1 * X1 + Y1 * Y1 ))
               C1 = X2 * X1
               S1 = X2 * Y1
               C2 = C1 * C1 - S1 * S1
               S2 = C1 * S1 + C1 * S1
               C3 = C2 * C1 - S2 * S1
               S3 = C2 * S1 + S2 * C1
 8510          CONTINUE
               DO 8540 K0 = KK, N2, BLK
                  K1 = K0 + SPAN
                  K2 = K1 + SPAN
                  K3 = K2 + SPAN
                  IF ( KK .GT. 1 ) GO TO 8520
                    X1 = Z(K1)
```

```
                Y1 = Z(K1+1)
                X2 = Z(K2)
                Y2 = Z(K2+1)
                X3 = Z(K3)
                Y3 = Z(K3+1)
                GO TO 8530
8520            CONTINUE
                X1 = Z(K1) * C2 - Z(K1+1) * S2
                Y1 = Z(K1) * S2 + Z(K1+1) * C2
                X2 = Z(K2) * C1 - Z(K2+1) * S1
                Y2 = Z(K2) * S1 + Z(K2+1) * C1
                X3 = Z(K3) * C3 - Z(K3+1) * S3
                Y3 = Z(K3) * S3 + Z(K3+1) * C3
8530            CONTINUE
                Z(K3) = Z(K0) - X1 + Y2 - Y3
                Z(K2) = Z(K0) + X1 - X2 - X3
                Z(K1) = Z(K0) - X1 - Y2 + Y3
                Z(K0) = Z(K0) + X1 + X2 + X3
                Z(K3+1) = Z(K0+1) - Y1 - X2 + X3
                Z(K2+1) = Z(K0+1) + Y1 - Y2 - Y3
                Z(K1+1) = Z(K0+1) - Y1 + X2 - X3
                Z(K0+1) = Z(K0+1) + Y1 + Y2 + Y3
8540         CONTINUE
8550     CONTINUE
         SPAN = BLK
         GO TO 8500
8560 CONTINUE
     IF ( SIGN .GE. 0 ) RETURN
       DO 8600 KK = 1, N2, 2
8600       Z(KK+1) = -Z(KK+1)
     RETURN
     END


C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFS (N, Z, SIGN)
      DOUBLE PRECISION Z(n)
      INTEGER N, SIGN
      INTEGER K0, K1, ND2
      DOUBLE PRECISION PI, CE, C1, SE, S1, SF, X1, X2, Y1, Y2
      DATA PI /3.1415 92653 58979 324/
      ND2 = IABS(N) / 2
      IF (ND2 + ND2 .NE. N) RETURN
      SF = 1. / FLOAT(ND2)
      X1 = PI * SF
      IF (SIGN .GE. 0) GO TO 8100
       SF = .5
       S1 = 1.
       Z(2) = SF * (Z(1) - Z(N+1))
       Z(1) = SF * (Z(1) + Z(N+1))
       X1 = -X1
       GO TO 8110
8100 CONTINUE
       Z(N+1) = SF * (Z(1) - Z(2))
       Z(N+2) = 0.0
       Z(1) = SF * (Z(1) + Z(2))
       Z(2) = 0.0
       IF (MOD(ND2,2) .NE. 0) GO TO 8105
         Z(ND2+1) = SF * Z(ND2+1)
         Z(ND2+2) = SF * Z(ND2+2)
8105   CONTINUE
       SF = .5 * SF
       S1 = -1.
8110 CONTINUE
     C1 = 0.0
     CE = COS(X1)
     SE = SIN(X1)
     K1 = N - 1
     K0 = 3
8200 IF (K0 .GE. K1) RETURN
       X1 = CE * C1 - SE * S1
       Y1 = CE * S1 + SE * C1
       X2 = .5 * (3. - (X1 * X1 + Y1 * Y1))
       C1 = X1 * X2
       S1 = Y1 * X2
       X1 = Z(K0) - Z(K1)
       Y1 = Z(K0+1) + Z(K1+1)
       X2 = C1 * X1 - S1 * Y1
       Y2 = C1 * Y1 + S1 * X1
       X1 = Z(K0) + Z(K1)
       Y1 = Z(K0+1) - Z(K1+1)
       Z(K0) = SF * (X1 + X2)
       Z(K0+1) = SF * (Y1 + Y2)
       Z(K1) = SF * (X1 - X2)
       Z(K1+1) = SF * (Y2 -Y1)
       K1 = K1 - 2
       K0 = K0 + 2
       GO TO 8200
     END
```

This program allows the calculation of the dipolar dependence on coherence transfer (instead of time domain, the x axis is the dipolar coupling constant)

**PROGRAM D-POWDER**

```
      Implicit double precision (a-h, o-z)
      INTEGER *2 FL
      REAL TIME, RSTEP,RMAX,RMIN
      DOUBLE PRECISION PI, HBAR, APAS
      CHARACTER *30 FID, FFTOUT
      CHARACTER *1 STATE, SIGNI, FFTYN
      DATA PI /3.1415 92653 58979 324/
C     Planck's constant in cgs (erg sec rad(-1))
      DATA HBAR /1.0545887E-27/

      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
5     PRINT *, 'Calculations are in cgs.'
      PRINT *, ''
      PRINT *,''

      PRINT *, 'Input the eqn you want to calc (A,B,C,D,E,F,G).'
      READ (*,'(A)') STATE

      IF ((STATE.EQ.'A').OR.(STATE.EQ.'B').OR.(STATE.EQ.'C').OR.
     &(STATE.EQ.'D').OR.(STATE.EQ.'E').OR.(STATE.EQ.'F').OR.
     &(STATE.EQ.'G')) THEN
      STATE=STATE
      ELSE
      PRINT *, 'You must enter a state of A, B, C, D, E, F or G.
     &Please try again.'
      GOTO 5
      ENDIF

      PRINT *, 'Is this equation a negative? -A, -B, etc? (Y or
N)'
      READ (*,'(A)') SIGNI

      IF ((SIGNI.EQ.'Y').OR.(SIGNI.EQ.'y')) THEN
      SIGN= -1.0
      ELSE
         SIGN=1.0
      ENDIF
      PRINT *, 'What is the name for the data file?'
      READ (*, '(A)') FID
      OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')

C  Parameter Input

      PRINT *, 'Parameters'
      PRINT *, ''
      PRINT *, 'Spinning Angle in degrees? (MA = 54.73561)'
      READ *, BNGLE
      PRINT *, 'Spinning Frequency? (hertz)'
      READ *, OMEGA
      TSTME = 1/OMEGA * 1000
      PRINT *, 'Stroboscopic sample time? (millisecs).'
      PRINT *, 'For MASS at a frequency of ',OMEGA,' Hz,
suggested',
     & 'value is ',TSTME,' ms, or integer multpiles such as
',2*TSTME,
     & ' ms, or ', 3*TSTME, ' ms, etc.'
```

```
      READ *, STIME
      PRINT *, 'Value of Gamma for homonuclear pair?'
      PRINT *, '(C13 = 6727.0, H = 26750.0, in rad sec(-1) G(-
1))'
      READ  *, GAMMA
      PRINT *, 'Value of J for this pair (in Hz)?'
      READ *, JJJ
      PRINT *, 'Initial dipolar coefficient between nuclei? (in
Hz)'
      READ *, DINIT
      PRINT *, 'Maximum dipolar coefficient between nuclei (in
Hz)'
      READ *, DMAX
      PRINT *, 'Increments of dipolar constant?'
      READ *, DSTEP
      PRINT *, 'Constant Time value? (ms)'
      READ *, TIME
      PRINT *, 'Number of Steps in Theta/Phi Summation ( > w/>
Time)'
      READ *, N

      TIME=TIME*1.E-03


C File Opening Units

C  UNIT CONVERSIONS AND ETC (cgs system)


       DCUR = DINIT


C  BNGLE is the spinning angle of the rotor sample converted from
deg to rad

      ANGLE = BNGLE * PI/180.

C  STIME is the stroboscopic sampling time converted from ms to
s.

      STIME = STIME * 1.E-3

C  OMEGAR is the spinning frequency, here converted to
radians/sec.
C  SOMEGA is STIME*OMEGAR is the phase due to sampling at
different
C  times in the rotor rotation.

      OMEGAR = OMEGA * 2. * PI
      SOMEGA = OMEGAR * STIME

C  TNORM is the normalization constant so when we add up
integration
C  in the do loops, it normalizes to a max magnitude (amplitude)
of 1.

      TNORM = 1./(N**2)
      OUTPUT = 1

C  APAS is the constant in front of the dipolar hamiltonian...
C  gamma1*gamma2*hbar/R^3...since gamma1=gamma2, it's just
squared.
C  Note: in powders, there is no factor of 1/2 or any other
multiplier
C  with this simple dipolar constant...except here, 1/pi to
balance out
C  the PI that must be inside the trig functions in the loops (or
I
C  could have multiplied J by PI...same thing, but this way I can
C  keep J and D inside the same parenthesis by dividing D by pi).
C (Note APAS is negative, like D is negative)

59       APAS=DCUR


C Pieces of the spatial part of the Hamiltonian which determines
the
C frequencies we will integrate over.
C I broke it up to make spot-checks easier and to cut down
calculation
C in the loops.
C This is a simple Riemann sum. There are likely more accurate
algorithms,
C but this seems to work okay.
C REMINDER:
C The angle A is the angle THETA in the lab frame. The angle B is
PHI.
C ANGLE is the angle of the goniometer. STIME is the 'sample
time' which
C we multiply by the MAS spinning speed OMEGAR.


          XNGLE1 = SIN(ANGLE)**2
           XNGLE4 = SIN(2*ANGLE)
```

```
          XNGLE7 = 3*(COS(ANGLE)**2)


89     CONTINUE

C ---------------------- CALCULATIONS FOR EQN A -----------------

            IF (STATE.EQ.'A') THEN

C  Calculational doloop. One nested for theta, the other for phi.
C  A is theta, B is phi. The idea is to generate one frequency
C  over a range of frequencies from theta=0...pi and phi=0...2*pi
C  and then for each possible f requency, calculate ALL the time
domain.
C  Then, you add each time point for each frequency into the
array
C  'Result1' and 'Result2' where the arrays are indexed by the
time
C   points.   This sum of all possible frequencies will result in
a FID.

C The J=N  assures that the  J LOOP is the same as N for PHI (B)
as
C for THETA (A) I made them seperate so other goofy things could
be
C done later  (like experimentation with integration steps over
PHI).

          J = N

           DO 138  I = 1, N, 1
               A = PI * I/N
                 DO 238 K = 1, J, 1
                      B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*(SOMEGA + B))
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

C Note my 'APAS' here as above is in CGS.
          TNS=-APAS*(TNS1 - TNS2 + TNS3)

C  CONDITIONAL LOGIC FOR STATES IN TIME-CALCULATION LOOP
C  THIS LOOP INCREMENTS TIME WITH THE NEW FREQUENCY, AND
C  CALCULATES THE VALUE OF <STATE> FOR EACH TIME INTERVAL

C              Calculation for <eqn A>

            TIMRE=0.25*PI*SIN(A)*TNORM*
     &(COS(1.5*TNS*TIME*PI)+COS((0.5*TNS + JJJ)*PI*TIME))
            TIMERS = TIMRE+TIMERS

238          CONTINUE
138        CONTINUE


C ----------------- CALC FOR EQN B --------------------------

          ELSE IF (STATE.EQ.'B') THEN

          J = N

           DO 139  I = 1, N, 1
               A = PI * I/N
                 DO 239 K = 1, J, 1
                      B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*(SOMEGA + B))
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=-APAS*(TNS1 - TNS2 + TNS3)


            TIMRE=0.25*PI*SIN(A)*TNORM*SIGN*
     &(COS(3/2*TNS*PI*TIME)-COS((0.5*TNS+JJJ)*PI*TIME))
            TIMERS=TIMRE+TIMERS

239          CONTINUE
139        CONTINUE
```

```
C--------------- CALC FOR EQN C ---------------------

         ELSE IF (STATE.EQ.'C') THEN

          J = N

           DO 141  I = 1, N, 1
               A = PI * I/N
               DO 241 K = 1, J, 1
                     B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*SOMEGA + 2.*B)
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

           TNS=-APAS*(TNS1 - TNS2 + TNS3)

C              Calculation for <eqn C>

            TIMRE=0.25*PI*SIN(A)*TNORM*
     &(1+COS((JJJ-TNS)*PI*TIME))

           TIMERS=TIMRE+TIMERS

241          CONTINUE
141         CONTINUE



C--------------- CALC FOR EQN D ---------------------

         ELSE IF (STATE.EQ.'D') THEN

          J = N

           DO 142  I = 1, N, 1
               A = PI * I/N
               DO 242 K = 1, J, 1
                    B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*SOMEGA + 2.*B)
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=-APAS*(TNS1 - TNS2 + TNS3)


C              Calculation for <eqn D>

           TIMRE=0.25*PI*SIN(A)*TNORM*
     &(1-COS((JJJ-TNS)*PI*TIME))
           TIMERS = TIMERS+TIMRE

242          CONTINUE
142         CONTINUE



C--------------- CALC FOR EQN E ---------------------

         ELSE IF (STATE.EQ.'E') THEN

          J = N

           DO 143  I = 1, N, 1
               A = PI * I/N
                DO 243 K = 1, J, 1
                     B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*SOMEGA + 2.*B)
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
```

```
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=-APAS*(TNS1 - TNS2 + TNS3)


C              Calculation for <eqn E>

           TIMRE=0.25*PI*SIN(A)*TNORM*SIGN*
     &(SIN((0.5*TNS+JJJ)*PI*TIME) + SIN(1.5*TNS*PI*TIME))
          TIMERS=TIMRE+TIMERS

243         CONTINUE
143       CONTINUE



C--------------- CALC FOR EQN F  ---------------------

        ELSE IF (STATE.EQ.'F') THEN

         J = N

          DO 145  I = 1, N, 1
              A = PI * I/N
                DO 245 K = 1, J, 1
                     B = 2 * PI * K/J

          XNGLE2 = SIN(A)**2
          XNGLE3 = COS(2.*SOMEGA + 2.*B)
          XNGLE5 = SIN(2.*A)
          XNGLE6 = COS(SOMEGA+B)
          XNGLE8 = 3.*(COS(A))**2

          TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
          TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=-APAS*(TNS1 - TNS2 + TNS3)


C              Calculation for <eqn F>

           TIMRE= 0.25*PI*SIN(A)*TNORM*SIGN*
     &(SIN((0.5*TNS+JJJ)*PI*TIME) - SIN(1.5*TNS*PI*TIME))
            TIMERS=TIMRE+TIMERS

245         CONTINUE
145       CONTINUE


C--------------- CALC FOR EQN G  ---------------------

        ELSE IF (STATE.EQ.'G') THEN

         J = N

          DO 147  I = 1, N, 1
              A = PI * I/N
                DO 247 K = 1, J, 1
                     B = 2 * PI * K/J

          XNGLE2 = SIN(A)**2
          XNGLE3 = COS(2.*SOMEGA + 2.*B)
          XNGLE5 = SIN(2.*A)
          XNGLE6 = COS(SOMEGA+B)
          XNGLE8 = 3.*(COS(A))**2

          TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
          TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=-APAS*(TNS1 - TNS2 + TNS3)


C              Calculation for <eqn G>

       TIMRE=0.25*PI*SIN(A)*SIGN*TNORM*SIN((JJJ-TNS)*PI*TIME)
        TIMERS=TIMRE+TIMERS

247         CONTINUE
147       CONTINUE


         ENDIF

C Some useful write-tos to the output file.
        IF (OUTPUT.EQ.1) THEN
       WRITE (4,*)' RESULTS FOR: ', STATE
       WRITE (4,*)' --------------------------'
       WRITE (4,*)' '
       WRITE (4,*)' No. of steps in integration: ', N
       WRITE (4,*)' Gamma for homonuclear pair: ', GAMMA
      WRITE (4,*)' No. of data points reporting:', M
```

```
       WRITE (4,*)' Spinning Angle: ', ANGLE
       WRITE (4,*)' Stroboscopic Sample Time ', STIME
       WRITE (4,*)' Value of J: ' , JJJ
       WRITE (4,*)' Spinning Speed: ' , OMEGAR
       WRITE (4,*)' Constant Time Value ' , TIME
       WRITE (4,*)' Value of D between ', DINIT, ' and ', DMAX
       WRITE (4,*)' ----------------'
       WRITE (4,*)' Evolution of EQN ', STATE
       WRITE (4,*)' ----------------'

       ENDIF

C  Write to file. Re-initialize 'Result'.

       OUTPUT=0
            WRITE(4,*) DCUR,  TIMERS
           TIMRE=0
            TIMERS=0

        DCUR = DCUR + DSTEP
        IF (DCUR.LT.DMAX) THEN
        GOTO 59
        ENDIF

        END



PROGRAM DRAMA

C      This program calculates the response of a two-spin
C      system to the dipolar Hamiltonian (Hd) when one of
C      the spins (the I spin) is perturbed. It can also
C      include isotropic J as well as D. It then graphs
C      the FID response of spin S as well as spin I to two
C      seperate files so you can plot them. This also
C      involves itself with FFT calculations. Comments provided
C      for more clue.  This program was verified with an
C      H-H calculation on Sz at 2.5 angstroms against
C      R. Bruschweiler's  review article, page 5, in
C      Progress in Nuclear Mag Resonance Spec, 32 (1998)
C      - Deanne Taylor, Sept, 1998.

       Implicit double precision (a-h, o-z)
       INTEGER *2 FL
       REAL TSTEP
       DOUBLE PRECISION PI,HBAR,APAS,TAU1,TAU2,RATIO,GAMMA
       CHARACTER *30 FID, FFTOUT
       CHARACTER *1 STATE, SIGNI, FFTYN
       DIMENSION  RES1(50000),  RESULT1(50000), CONFIG(100000)
       DIMENSION AT(32786), BT(16393), CT(16393)
       DATA  PI /3.14159265358979324/
C      Planck's constant in cgs (erg sec rad(-1))
       DATA HBAR /1.0545887E-27/

       PRINT *, ''
       PRINT *, ''
       PRINT *, 'This program calculates the average hamiltonian'
       PRINT *, 'response of a system starting as Ix, Iy, or Iz.'
       PRINT *, 'The Hd used is from Tycko\'s 1990 paper in CPL'
       PRINT *, 'Vol 173 on page 462.'
       PRINT *, ''
       PRINT *, ''
5      PRINT *, 'Calculations are in cgs.'
       PRINT *, ''
       PRINT *,''

       PRINT *, 'Input the eqn you want to calc (A,B,C,D or E).'
       READ (*,'(A)') STATE
       IF ((STATE.EQ.'A').OR.(STATE.EQ.'B').OR.(STATE.EQ.'C')
     &.OR.(STATE.EQ.'D').OR.(STATE.EQ.'E')) THEN
       STATE=STATE
       ELSE
       PRINT *, 'You must enter a state of A, B, C, or D."
     &Please try again.'
       GOTO 5
       ENDIF

      PRINT *, 'Is this equation a negative? -A, -B, etc? (Y or
N)'
      READ (*,'(A)') SIGNI

      IF ((SIGNI.EQ.'Y').OR.(SIGNI.EQ.'y')) THEN
      SIGN= -1.0
      ELSE
         SIGN=1.0
      ENDIF

C  Parameter Input
```

```
      PRINT *, 'What is the name for the data file?'
      READ (*, '(A)') FID
      OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')
      OPEN (UNIT=3, STATUS='UNKNOWN')
      OPEN (UNIT=88, STATUS='UNKNOWN')
       PRINT *, 'Rotor Spinning Speed (Hz)?'
      READ *, GNUR
      PRINT *, 'Ratio of inner evolution time to rotor period?'
      READ *, RATIO
      TAUR = 1/GNUR
      TAU2 = RATIO*TAUR
      TAU1 = (TAUR-TAU2)/2
      PRINT *, 'Value of Gamma for homonuclear pair?'
      PRINT *, '(C13 = 6727.0, H = 26750.0, in rad sec(-1) G(-
1))'
      READ  *, GAMMA
      PRINT *, 'Distance between nuclei? (in Angstroms)'
      READ *, R
      PRINT *, 'Number of steps in theta, phi integration?'
      READ *, N
      PRINT *, 'Do you want an FFT calculated?'
      READ *, FFTYN
      IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN
       PRINT *, 'Number of steps in time array? Power of 2, max
32784'
       READ *, M
       PRINT *, 'Name of FFT output file?'
       READ *, FFTOUT
      ELSE
       PRINT *, 'Number of steps in time array? (Even # up to
10,000)'
       READ *, M
      ENDIF
      PRINT *, '__', TAU1 , '__|_', TAU2 ,'_|__', TAU1 ,'__'
C File Opening Units

C  UNIT CONVERSIONS AND ETC (cgs system)

C  R is the internuclear distance converted into cm.

      R = R * 1.E-8

C  OMEGAR is the spinning frequency, here converted to
radians/sec.

      OMEGAR = GNUR * 2. * PI

C  TNORM is the normalization constant so when we add up
integration
C  in the do loops, it normalizes to a max magnitude (amplitude)
of 1.

      TNORM = 1./(N**2)

C  APAS is the constant in front of the dipolar hamiltonian...
C  gamma1*gamma2*hbar/R^3...since gamma1=gamma2, it's just
squared.
C  Note: in powders, there is no factor of 1/2 or any other
multiplier
C  with this simple dipolar constant...except here, 1/pi to
balance out
C  the PI that must be inside the trig functions in the loops (or
I
C  could have multiplied J by PI...same thing, but this way I can
C  keep J and D inside the same parenthesis by dividing D by pi).
C (Note APAS is negative, like D is negative)

      APAS=-(GAMMA**2)*HBAR*(1/R**3)

C TEST determines the measure of time scale (see below).

      TEST=ABS(3.*APAS/SQRT(6.))

      PRINT *, 'Value of spinning angle = ', ANGLE, 'radians.'
      PRINT *, 'Value of APAS:', APAS, ' rad/sec'
      PRINT *, '               ', APAS/(2.*PI), ' Hz'
      PRINT *, 'Distance in cm: ',R
      PRINT *, 'The gamma pair: ' , GAMMA

C     TIMESTEP CALCULATION using TEST (these seem to work well)

      IF (TEST.LT.1000.) THEN
             TSTEP=.00001
      ELSE IF (TEST.LT.5000.) THEN
             TSTEP=.00001
      ELSE IF (TEST.LT.10000.) THEN
             TSTEP=.00001
      ELSE IF (TEST.LT.20000.) THEN
             TSTEP=.00001
      ELSE IF (TEST.LT.50000.) THEN
             TSTEP=.00001
      ELSE
             TSTEP=.00001
      ENDIF
```

```
       PRINT *, 'Timesteps selected: ', TSTEP, ' seconds.'
       PRINT *, 'Your total data set will span ', TSTEP*M*1E03,
     &' milliseconds.'

C Pieces of the spatial part of the Hamiltonian which determines
the
C frequencies we will sum over

89       PRINT *, 'Now calculating FID for Equation ', STATE, '.'

C ---------------------------------------------------------------

       CST1 =  SQRT(2.)/(2.*PI)
        CST2 =  (1/(4.*PI))

C ---------------------- CALCULATIONS FOR EQN C -----------------

            IF (STATE.EQ.'C') THEN

          J = N

           DO 155  I = 1, N, 1
               A = PI * I/N

               DO 255 K = 1, J, 1
                     B = 2 * PI * K/J


      XNGL1=CST1*Sin(2.*A)*Cos(B)*Sin(TAU2*OMEGAR/2.)
      XNGL2=CST2*COS(2.*B)*SIN(A)**2 * SIN(TAU2*OMEGAR)
C Note my 'APAS' here as above is in CGS.

         TNS=3.*APAS/SQRT(6.)*(XNGL1+XNGL2)

C  CONDITIONAL LOGIC FOR STATES IN TIME-CALCULATION LOOP
C  THIS LOOP INCREMENTS TIME WITH THE NEW FREQUENCY, AND
C  CALCULATES THE VALUE OF <STATE> FOR EACH TIME INTERVAL

C               Calculation for <eqn C>
C               NOTE THAT IN TYCKO'S MATH, THERE IS
C               NO  Sz evolution FROM Iz under
C               MASS and RECOVERED DIPOLAR HAMILTONIAN

         DO 30 NTIME = 0, M, 1
          DELTAT = NTIME * TSTEP
          RES1(NTIME)=0.5*PI*SIN(A)*TNORM*COS(0.5*TNS*DELTAT)
          RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)

30       CONTINUE

255         CONTINUE
155      CONTINUE

C  The math for the frequency calcs are different in the
following two
C  elseifs  for Ix and Iy, but otherwise, the routines are the
same.
C  See notes under Iz calc section.


C ------------------ CALCULATION FOR EQN A ---------------------

            ELSE IF (STATE.EQ.'A') THEN

           DO 400  I = 1, N, 1
            A = PI * I/N
              DO 500 K = 1, N, 1
                B = 2 * PI * K/N

      XNGL1=CST1*Sin(2.*A)*Cos(B)*Sin(TAU2*OMEGAR/2.)
      XNGL2=CST2*COS(2.*B)*SIN(A)**2 * SIN(TAU2*OMEGAR)

C Note my 'APAS' here as above is in CGS.

         TNS=3.*APAS/SQRT(6.)*(XNGL1+XNGL2)

C          Calculation for <A>

      DO 50 NTIME = 0, M, 1
      DELTAT = NTIME * TSTEP
      RES1(NTIME)=0.25*TNORM*SIN(A)*PI*(1+COS(TNS*DELTAT))
      RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
50    CONTINUE

500    CONTINUE
400    CONTINUE

C--------------- CALC FOR EQN B ---------------------

         ELSE IF (STATE.EQ.'B') THEN

          J = N

           DO 141  I = 1, N, 1
               A = PI * I/N
```

```
              DO 241 K = 1, J, 1
                    B = 2 * PI * K/J

      XNGL1=CST1*Sin(2.*A)*Cos(B)*Sin(TAU2*OMEGAR/2.)
      XNGL2=CST2*COS(2.*B)*SIN(A)**2 * SIN(TAU2*OMEGAR)

C Note my 'APAS' here as above is in CGS.

         TNS=3.*APAS/SQRT(6.)*(XNGL1+XNGL2)


C             Calculation for <eqn B>

       DO 14 MTIME = 0, M, 1
         DELTAT = MTIME * TSTEP
         RES1(MTIME)= 0.25*TNORM*SIN(A)*PI*SIGN*
     &(1-COS(TNS*DELTAT))
         RESULT1(MTIME) = RESULT1(MTIME) + RES1(MTIME)

14       CONTINUE

241          CONTINUE
141         CONTINUE

C ---------------------- CALCULATIONS FOR EQN D -----------------
             ELSE IF (STATE.EQ.'D') THEN
           J = N
            DO 159  I = 1, N, 1
                A = PI * I/N
                DO 160 K = 1, J, 1
                   B = 2 * PI * K/J
      XNGL1=CST1*Sin(2.*A)*Cos(B)*Sin(TAU2*OMEGAR/2.)
      XNGL2=CST2*COS(2.*B)*SIN(A)**2 * SIN(TAU2*OMEGAR)

         TNS=3.*APAS/SQRT(6.)*(XNGL1+XNGL2)

C              Calculation for <eqn D>

       DO 301 MTIME = 0, M, 1
         DELTAT = MTIME * TSTEP
         RES1(MTIME)= 0.5*TNORM*SIN(A)*PI*SIGN*
     &(SIN(0.5*TNS*DELTAT))
         RESULT1(MTIME) = RESULT1(MTIME) + RES1(MTIME)

301    CONTINUE

160         CONTINUE
159      CONTINUE


C ---------------------- CALCULATIONS FOR EQN E -----------------

             ELSE IF (STATE.EQ.'E') THEN

           J = N

            DO 169  I = 1, N, 1
               A = PI * I/N

                DO 170 K = 1, J, 1
                      B = 2 * PI * K/J

      XNGL1=CST1*Sin(2.*A)*Cos(B)*Sin(TAU2*OMEGAR/2.)
      XNGL2=CST2*COS(2.*B)*SIN(A)**2 * SIN(TAU2*OMEGAR)

C Note my 'APAS' here as above is in CGS.

         TNS=3.*APAS/SQRT(6.)*(XNGL1+XNGL2)


C              Calculation for <eqn E>

         DO 302 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=0.25*PI*SIN(A)*SIGN*TNORM*
     &SIN(TNS*DELTAT)
           RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
302       CONTINUE

170        CONTINUE
169      CONTINUE

C  The math for the frequency calcs are different in the
following two
C  elseifs  for Ix and Iy, but otherwise, the routines are the
same.
C  See notes under Iz calc section.


        ELSE
               PRINT *,'You need to specify a state to evolve.'
               GOTO 5

       ENDIF
```

```
C Some useful write-tos to the output file.

       WRITE (4,*)' RESULTS FOR: ', STATE
       WRITE (4,*)' --------------------------'
       WRITE (4,*)' '
       WRITE (4,*)' No. of steps in integration: ', N
       WRITE (4,*)' Gamma for homonuclear pair: ', GAMMA
       WRITE (4,*)' No. of data points reporting:', M
       WRITE (4,*)' No. of total data points:', M
       WRITE (4,*)' Internuclear Distance (angstroms): ', R*1E8
       WRITE (4,*)' RATIO inner evolution to rot period:', RATIO
       WRITE (4,*)' TAU1:' , TAU1, 'TAU2:' , TAU2
       WRITE (4,*)' Spinning Speed: ' , GNUR ,  ' Hz'
       WRITE (4,*)' Value of D: ', APAS/(2.*PI),  'Hz'
       WRITE (4,*)' Value of D: ', APAS, 'rad/sec'
       WRITE (4,*)' ----------------'
       WRITE (4,*)' Evolution of EQN ', STATE , ' in time steps
of',
     &TSTEP*1E03, ' milliseconds'
       WRITE (4,*)' ----------------'

       PRINT *, 'Writing to FID file...'
       DO 111 L=0, M, 1
            WRITE(4,*)  L*1E03*TSTEP,  '   ' , RESULT1(L)
111      CONTINUE

          WRITE(4,*)'-9999'

9999   IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN

        DO 107 I=1, M, 1
        CONFIG(I)=RESULT1(I)
107     CONTINUE

        DO 109  I=1, M, 1
       WRITE (3,*) I*1E03*TSTEP, '    ', CONFIG(I)
109     CONTINUE
       dist=r
       CALL FFT(CONFIG,M,FFTOUT,TSTEP,STATE,GAMMA,
     &N,APAS, RATIO, TAU1, TAU2, GNUR, dist)
       ENDIF
       END

      SUBROUTINE FFT (AT,M,FFTOUT,TSTEP,STATE,GAMMA,
     &N,APAS,RATIO, TAU1, TAU2, GNUR, dist)
      REAL TSTEP
      DOUBLE PRECISION
PI,HBAR,APAS,TAU1,TAU2,RATIO,GAMMA,GNUR,dist
      CHARACTER *1 STATE, SIGNI, FFTYN
      INTEGER I, J
      DOUBLE PRECISION AT(M+2),BT(M/2+1),CT(M/2+1),DT(M/2+1)
      CHARACTER * 30 FFTOUT
      DATA PI /3.1415 92653 58979 324/
      OPEN(5, FILE=FFTOUT)
      CLOSE(4)
      CALL SFFT2(M/2,AT,1)
      CALL SFS(M,AT,1)
      PRINT *, 'Writing to FFT file...'
      WRITE (5,*), 'FFT CALCULATION FOR EQN ', STATE
       WRITE (5,*)' --------------------------'
       WRITE (5,*)' '
       WRITE (5,*)' No. of steps in integration: ', N
       WRITE (5,*)' Gamma for homonuclear pair: ', GAMMA
       WRITE (5,*)' No. of data points reporting:', M
       WRITE (5,*)' Internuclear Distance (angstroms):',dist*1E8
       WRITE (5,*)' RATIO inner evolution to rot period:', RATIO
       WRITE (5,*)' TAU1:' , TAU1, 'TAU2:' , TAU2
       WRITE (5,*)' Spinning Speed: ' , GNUR ,  ' Hz'
       WRITE (4,*)' Value of D: ', APAS/(2.*PI),  'Hz'
       WRITE (4,*)' Value of D: ', APAS, 'rad/sec'
       WRITE (5,*)' ----------------'
       WRITE (5,*)' FFT of EQN ', STATE , ' in freq steps of',
     &1/(M*1E03*TSTEP), ' milliseconds'
       WRITE (5,*)' ----------------'


      DO 200 J=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(J)=AT(M-(2*J-1))
200   CONTINUE
      BT(M/2)=0
      DO 201 J=1, M/2
      DT(J)=BT(J-1)
      WRITE(5,*), (J-(M/2+1))/(M*TSTEP), DT(J)*M
201   CONTINUE

      DO 202 I=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(I)=AT(2*I-1)
```

```
      CT(I)=AT(2*I)
      WRITE(88,*),CT(I)
202   CONTINUE

      BT(M/2 +1)=0.

      DO  203 I=1, M/2
      DT(I)=BT(I+1)
      WRITE(5,*), I/(M*TSTEP), DT(I)*M
203   CONTINUE

C Finish off with an end tag for the data output (for perl script
purposes)

      WRITE(5,*), '-9999'
      RETURN
      END


C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFFT2 (N,Z,SIGN)
      double precision Z(N)
      INTEGER N, SIGN
      INTEGER JBH, JBL, KR, LOG2
      INTEGER BLK, KK, K0, K1, K2, K3, N2, SPAN
      DOUBLE PRECISION CE, C1, C2, C3, SE, S1, S2, S3, PIOV2,
THETA
      DOUBLE PRECISION  X1, X2, X3, Y1, Y2, Y3
      DATA PIOV2/1.5707 96326 79489 662/
      N2 = 1
      LOG2 = 0
 8100    LOG2 = LOG2 + 1
         N2 = N2 + N2
         IF ( N2 .LT. N )  GO TO 8100
      IF ( N2 .GT. N ) RETURN
      N2 = N2 + N2 - 1
      IF ( SIGN .GE. 0 ) GO TO 8210
       DO 8200 KK = 1, N2, 2
 8200      Z(KK+1) = -Z(KK+1)
 8210 CONTINUE
      JBL = 2
      JBH = N
 8300 IF ( JBH .LE. JBL ) GO TO 8340
         SPAN = JBL + JBL
         KR = 1
 8310        KK = KR + JBL
             KR = KR + JBH
             BLK = KR - JBL
             DO 8330 K1 = KK, BLK, SPAN
                K3 = K1 + JBL - 2
                DO 8320 K0 = K1, K3, 2
                    X1 = Z(K0)
                    Z(K0) = Z(KR)
                    Z(KR) = X1
                    Y1 = Z(K0+1)
                    Z(K0+1) = Z(KR+1)
                    Z(KR+1) = Y1
 8320               KR = KR + 2
                KR = KR + JBL
 8330        CONTINUE
             IF ( KR .LT. N2 ) GO TO 8310
          JBH = JBH / 2
          JBL = JBL + JBL
          GO TO 8300
 8340 CONTINUE
      IF ( MOD(LOG2,2) .NE. 0 ) GO TO 8400
       THETA = PIOV2
       SPAN = 2
       GO TO 8420
 8400 CONTINUE
       DO 8410 K0 = 1, N2, 4
          K1 = K0 + 2
          X1 = Z(K0)
          Z(K0) = X1 + Z(K1)
          Z(K1) = X1 - Z(K1)
          Y1 = Z(K0+1)
          Z(K0+1) = Y1 + Z(K1+1)
 8410     Z(K1+1) = Y1 - Z(K1+1)
       THETA = PIOV2 + PIOV2
       SPAN = 4
 8420 CONTINUE
 8500 IF ( SPAN .GE. N2 ) GO TO 8560
         BLK = SPAN * 4
         IF ( SPAN .EQ. 2 ) GO TO 8505
           THETA = .25 * THETA
           SE = SIN(THETA)
           CE = COS(THETA)
           C1 = 1.0
           S1 = 0.0
 8505    CONTINUE
         DO 8550 KK = 1, SPAN, 2
             IF ( KK .EQ. 1 ) GO TO 8510
              X1 = C1 * CE - S1 * SE
```

```
            Y1 = C1 * SE + S1 * CE
            X2 = .5 * ( 3. - ( X1 * X1 + Y1 * Y1 ))
            C1 = X2 * X1
            S1 = X2 * Y1
            C2 = C1 * C1 - S1 * S1
            S2 = C1 * S1 + C1 * S1
            C3 = C2 * C1 - S2 * S1
            S3 = C2 * S1 + S2 * C1
8510       CONTINUE
           DO 8540 K0 = KK, N2, BLK
              K1 = K0 + SPAN
              K2 = K1 + SPAN
              K3 = K2 + SPAN
              IF ( KK .GT. 1 ) GO TO 8520
                X1 = Z(K1)
                Y1 = Z(K1+1)
                X2 = Z(K2)
                Y2 = Z(K2+1)
                X3 = Z(K3)
                Y3 = Z(K3+1)
                GO TO 8530
8520            CONTINUE
                X1 = Z(K1) * C2 - Z(K1+1) * S2
                Y1 = Z(K1) * S2 + Z(K1+1) * C2
                X2 = Z(K2) * C1 - Z(K2+1) * S1
                Y2 = Z(K2) * S1 + Z(K2+1) * C1
                X3 = Z(K3) * C3 - Z(K3+1) * S3
                Y3 = Z(K3) * S3 + Z(K3+1) * C3
8530            CONTINUE
                Z(K3) = Z(K0) - X1 + Y2 - Y3
                Z(K2) = Z(K0) + X1 - X2 - X3
                Z(K1) = Z(K0) - X1 - Y2 + Y3
                Z(K0) = Z(K0) + X1 + X2 + X3
                Z(K3+1) = Z(K0+1) - Y1 - X2 + X3
                Z(K2+1) = Z(K0+1) + Y1 - Y2 - Y3
                Z(K1+1) = Z(K0+1) - Y1 + X2 - X3
                Z(K0+1) = Z(K0+1) + Y1 + Y2 + Y3
8540        CONTINUE
8550    CONTINUE
        SPAN = BLK
        GO TO 8500
8560 CONTINUE
     IF ( SIGN .GE. 0 ) RETURN
      DO 8600 KK = 1, N2, 2
8600       Z(KK+1) = -Z(KK+1)
     RETURN
     END
```

```
C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFS (N, Z, SIGN)
      DOUBLE PRECISION Z(n)
      INTEGER N, SIGN
      INTEGER K0, K1, ND2
      DOUBLE PRECISION PI, CE, C1, SE, S1, SF, X1, X2, Y1, Y2
      DATA PI /3.1415 92653 58979 324/
      ND2 = IABS(N) / 2
      IF (ND2 + ND2 .NE. N) RETURN
      SF = 1. / FLOAT(ND2)
      X1 = PI * SF
      IF (SIGN .GE. 0) GO TO 8100
        SF = .5
        S1 = 1.
        Z(2) = SF * (Z(1) - Z(N+1))
        Z(1) = SF * (Z(1) + Z(N+1))
        X1 = -X1
        GO TO 8110
 8100 CONTINUE
        Z(N+1) = SF * (Z(1) - Z(2))
        Z(N+2) = 0.0
        Z(1) = SF * (Z(1) + Z(2))
        Z(2) = 0.0
        IF (MOD(ND2,2) .NE. 0) GO TO 8105
          Z(ND2+1) = SF * Z(ND2+1)
          Z(ND2+2) = SF * Z(ND2+2)
 8105   CONTINUE
        SF = .5 * SF
        S1 = -1.
 8110 CONTINUE
      C1 = 0.0
      CE = COS(X1)
      SE = SIN(X1)
      K1 = N - 1
      K0 = 3
 8200 IF (K0 .GE. K1) RETURN
         X1 = CE * C1 - SE * S1
         Y1 = CE * S1 + SE * C1
         X2 = .5 * (3. - (X1 * X1 + Y1 * Y1))
         C1 = X1 * X2
         S1 = Y1 * X2
         X1 = Z(K0) - Z(K1)
         Y1 = Z(K0+1) + Z(K1+1)
         X2 = C1 * X1 - S1 * Y1
```

```
        Y2 = C1 * Y1 + S1 * X1
        X1 = Z(K0) + Z(K1)
        Y1 = Z(K0+1) - Z(K1+1)
        Z(K0) = SF * (X1 + X2)
        Z(K0+1) = SF * (Y1 + Y2)
        Z(K1) = SF * (X1 - X2)
        Z(K1+1) = SF * (Y2 -Y1)
        K1 = K1 - 2
        K0 = K0 + 2
        GO TO 8200
      END

PROGRAM PDRAMA

       Implicit double precision (a-h, o-z)
       INTEGER *2 FL
       DOUBLE PRECISION PI, HBAR, APAS
       CHARACTER *30 FID, FFTOUT
       CHARACTER *1 STATE, SIGNI, FFTYN
       DATA PI /3.1415 92653 58979 324/
C      Planck's constant in cgs (erg sec rad(-1))
       DATA HBAR /1.0545887E-27/

5      PRINT *, 'Calculations are in cgs.'
       PRINT *, ''
       PRINT *,''
       PRINT *, 'Value of Gamma for homonuclear pair?'
       Print *, 'C13 = 6727.0, H=26750 in rad sec(-1) G(-1)'
       READ *, GAMMA
       PRINT *, 'What is the name for the data file?'
       READ (*, '(A)') FID
       PRINT *, 'Rotor Spinning Speed (Hz)?'
       READ *, GNUR
       PRINT *, 'Ratio of inner evolution time to rotor period?'
       READ *, RATIO
       PRINT *, 'Beginning time (ms)?'
       READ *, TIME
       PRINT *, 'Max time (milliseconds)'
       READ *, TMAX
       PRINT *, 'Increments of Time? (ms)'
       READ *, TSTEP
       PRINT *, 'Extreme distance value to begin search?
(angstroms)'
       READ *, RSTART
       PRINT *, 'Number of Steps in Theta/Phi Summation ( > w/>
Time)'
       READ *, N

6      PRINT *, "Tight (1) or Loose (2) search? (type 1 or 2)"
       READ *, NSRCH

       IF (NSRCH.EQ.1) THEN
       DCOARS=0.4
       RCOARS=0.05*1.E-08
       RFINE=0.0001*1.E-08
        ELSE IF (NSRCH.EQ.2) THEN
       DCOARS=0.44
       RCOARS=0.1*1.E-08
       RFINE=0.001*1.E-08
       ELSE
        PRINT *, 'Must input 1 or 2 for Tight or Loose Search'
       GOTO 6
       ENDIF

C File Opening Units

       OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')

C  UNIT CONVERSIONS AND ETC (cgs system)

       TAUR = 1./GNUR
       TAU2 = RATIO*TAUR
       TAU1 = (TAUR-TAU2)/2.
       TIME=TIME*1.E-3
       R=RSTART*1.E-8
       TMAX = TMAX * 1.E-3
       TSTEP = TSTEP*1.E-3

C  TNORM is the normalization constant so when we add up
integration
C  in the do loops, it normalizes to a max magnitude (amplitude)
of 1.

       TNORM = 1./(N**2)
       OMEGAR = GNUR * 2. * PI

        OUTPUT=1
        DIPS=0.01
        DIPOLD=0.
        DIPRE=0.
        GOTO 78

59     IF ((TIME.LE.TMAX).AND.(DIPS.LE.DIPOLD)) THEN
          WRITE (4,*) TIME*1E03, R*1.E08, DIPOLD
          PRINT *, TIME, TMAX,  R, DIPS, DIPOLD
```

```
          PRINT *, 'Incrementing TIME'
          OUTPUT=0
          TIME=TIME+TSTEP
         DIPOLD=0.
          R=RSTART*1E-08
          GOTO 78
        ELSE IF ((TIME.LE.TMAX).AND.
     &(DIPS.GT.DIPOLD)) THEN
         GOTO 60
        ELSE
         PRINT *, 'EOF'
         GOTO 9999
        ENDIF

60    IF (DIPS.LT.DCOARS) THEN
        PRINT *, 'COARSE  ',DIPS ,'for',DCOARS ,' was',  DIPOLD
         PRINT *, 'Incrementing R', R, ' to ', R-RCOARS
         DIPOLD=DIPS
         R=R-RCOARS
         GO TO 78
       ELSE IF (DIPS.GE.DCOARS) THEN
         PRINT *,'At FINE', DIPS ,'for ', DFINE ,' was', DIPOLD
         PRINT *, 'Incrementing R', R , 'to', R-RFINE
         DIPOLD=DIPS
         R=R-RFINE
         GO TO 78
       ENDIF


78        PRINT *, 'At 78, Proceding to next calc... '

              APAS = -(GAMMA**2)*HBAR*(1/R**3)
              DIPRE=0.
              DIPS=0.
C--------------------------------
       CST1 =  SQRT(2.)/(2.*PI)
        CST2 =  (1/(4.*PI))
C ---------------------- CALCULATIONS FOR EQN B -----------------

          J = N

           DO 141  I = 1, N, 1
              A = PI * I/N
              DO 241 K = 1, J, 1
                    B = 2 * PI * K/J

      FNT=0.25*PI*SIN(A)*TNORM
      XNGL1=CST1*Sin(2.*A)*Cos(B)*Sin(TAU2*OMEGAR/2.)
      XNGL2=CST2*COS(2.*B)*SIN(A)**2 * SIN(TAU2*OMEGAR)
      TNS=3.*APAS/SQRT(6.)*(XNGL1+XNGL2)

      DIPRE=FNT*(1-COS(TNS*TIME))
      DIPS=DIPS+DIPRE

241          CONTINUE
141         CONTINUE


529    IF (OUTPUT.EQ.1) THEN
        WRITE (4,*)' Maximum Search Results'
        WRITE (4,*)' --------------------------'
        WRITE (4,*)' Calculations of Distance(A),Time(ms), <D> '
        WRITE (4,*)' Ratio of TAU/TAUR: ', RATIO
        WRITE (4,*)' Spinning Frequency (Hz)', GNUR
        WRITE (4,*)' Gamma for homonuclear pair: ', GAMMA
        WRITE (4,*)' Value of D: ', APAS
        WRITE (4,*)' ----------------'
        WRITE (4,*)' ----------------'

        ENDIF

           OUTPUT=0
           GO TO 59
C     Write to file. Re-initialize 'Result'.

9999        END
```

**PROGRAM RPOWDER**

```
C      This program calculates the response of a two-spin
C      system to the dipolar Hamiltonian (Hd) when one of
C      the spins (the I spin) is perturbed. It can also
C      include isotropic J as well as D. It then graphs
C      the FID response of spin S as well as spin I to two
C      seperate files so you can plot them. This also
C      involves itself with FFT calculations. Comments provided
C      for more clue.  This program was verified with an
C      H-H calculation on Sz at 2.5 angstroms against
C      R. Bruschweiler's  review article, page 5, in
C      Progress in Nuclear Mag Resonance Spec, 32 (1998)
C      - Deanne Taylor, Sept, 1998.

       Implicit double precision (a-h, o-z)
       INTEGER *2 FL
```

```
      REAL TIME, RSTEP,RMAX,RMIN
      DOUBLE PRECISION PI, HBAR, APAS
      CHARACTER *30 FID, FFTOUT
      CHARACTER *1 STATE, SIGNI, FFTYN
      DATA PI /3.1415 92653 58979 324/
C     Planck's constant in cgs (erg sec rad(-1))
      DATA HBAR /1.0545887E-27/

      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
5     PRINT *, 'Calculations are in cgs.'
      PRINT *, ''
      PRINT *,''

      PRINT *, 'Input the eqn you want to calc (A,B,C,D,E,F,G).'
      READ (*,'(A)') STATE

      IF ((STATE.EQ.'A').OR.(STATE.EQ.'B').OR.(STATE.EQ.'C').OR.
     &(STATE.EQ.'D').OR.(STATE.EQ.'E').OR.(STATE.EQ.'F').OR.
     &(STATE.EQ.'G')) THEN
        STATE=STATE
        ELSE
        PRINT *, 'You must enter a state of A, B, C, D, E, F or G.
     &Please try again.'
        GOTO 5
        ENDIF

      PRINT *, 'Is this equation a negative? -A, -B, etc? (Y or
N)'
      READ (*,'(A)') SIGNI

      IF ((SIGNI.EQ.'Y').OR.(SIGNI.EQ.'y')) THEN
      SIGN= -1.0
      ELSE
         SIGN=1.0
      ENDIF
       PRINT *, 'What is the name for the data file?'
       READ (*, '(A)') FID
       OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')

C  Parameter Input

      PRINT *, 'Parameters'
      PRINT *, ''
      PRINT *, 'Spinning Angle in degrees? (MA = 54.73561)'
      READ *, BNGLE
      PRINT *, 'Spinning Frequency? (hertz)'
      READ *, OMEGA
      TSTME = 1/OMEGA * 1000
      PRINT *, 'Stroboscopic sample time? (millisecs).'
      PRINT *, 'For MASS at a frequency of ',OMEGA,' Hz,
suggested',
     & 'value is ',TSTME,' ms, or integer multpiles such as
',2*TSTME,
     & ' ms, or ', 3*TSTME, ' ms, etc.'
      READ *, STIME
      PRINT *, 'Value of Gamma for homonuclear pair?'
      PRINT *, '(C13 = 6727.0, H = 26750.0, in rad sec(-1) G(-
1))'
      READ  *, GAMMA
      PRINT *, 'Value of J for this pair (in Hz)?'
      READ *, JJJ
      PRINT *, 'Initial dipolar coupling constant? (in Hz)'
      READ *,   DIPINI
      PRINT *, 'Max dipolar coupling constant? (Hz)'
      READ *,  DIPMAX
      PRINT *, "Number of sample steps for dipolar axis?"
      READ *, DIPQRY
      PRINT *, 'Constant Time value? (ms)'
      READ *, TIME
      PRINT *, 'Number of Steps in Theta/Phi Summation ( > w/>
Time)'
      READ *, N
      TIME=TIME*1.E-03


C File Opening Units

C  UNIT CONVERSIONS AND ETC (cgs system)

C  R is the internuclear distance converted into cm.
      DIPVAL=DIPINI
      DIPSTP=(DIPMAX-DIPINI)/DIPQRY

C  BNGLE is the spinning angle of the rotor sample converted from
deg to rad

      ANGLE = BNGLE * PI/180.

C  STIME is the stroboscopic sampling time converted from ms to
s.
```

```
      STIME = STIME * 1.E-3

C  OMEGAR is the spinning frequency, here converted to
radians/sec.
C  SOMEGA is STIME*OMEGAR is the phase due to sampling at
different
C  times in the rotor rotation.

      OMEGAR = OMEGA * 2. * PI
      SOMEGA = OMEGAR * STIME

C  TNORM is the normalization constant so when we add up
integration
C  in the do loops, it normalizes to a max magnitude (amplitude)
of 1.

      TNORM = 1./(N**2)
      OUTPUT = 1

C  APAS is the constant in front of the dipolar hamiltonian...
C  gamma1*gamma2*hbar/R^3...since gamma1=gamma2, it's just
squared.
C  Note: in powders, there is no factor of 1/2 or any other
multiplier
C  with this simple dipolar constant...except here, 1/pi to
balance out
C  the PI that must be inside the trig functions in the loops (or
I
C  could have multiplied J by PI...same thing, but this way I can
C  keep J and D inside the same parenthesis by dividing D by pi).
C (Note APAS is negative, like D is negative)

59      APAS=DIPVAL*2*PI


C Pieces of the spatial part of the Hamiltonian which determines
the
C frequencies we will integrate over.
C I broke it up to make spot-checks easier and to cut down
calculation
C in the loops.
C This is a simple Riemann sum. There are likely more accurate
algorithms,
C but this seems to work okay.
C REMINDER:
C The angle A is the angle THETA in the lab frame. The angle B is
PHI.
C ANGLE is the angle of the goniometer. STIME is the 'sample
time' which
C we multiply by the MAS spinning speed OMEGAR.


          XNGLE1 = SIN(ANGLE)**2
           XNGLE4 = SIN(2*ANGLE)
           XNGLE7 = 3*(COS(ANGLE)**2)


89     CONTINUE

C ---------------------- CALCULATIONS FOR EQN A -----------------

             IF (STATE.EQ.'A') THEN

C  Calculational doloop. One nested for theta, the other for phi.
C  A is theta, B is phi. The idea is to generate one frequency
C  over a range of frequencies from theta=0...pi and phi=0...2*pi
C  and then for each possible f requency, calculate ALL the time
domain.
C  Then, you add each time point for each frequency into the
array
C  'Result1' and 'Result2' where the arrays are indexed by the
time
C   points.   This sum of all possible frequencies will result in
a FID.

C The J=N  assures that the  J LOOP is the same as N for PHI (B)
as
C for THETA (A) I made them seperate so other goofy things could
be
C done later  (like experimentation with integration steps over
PHI).

          J = N

           DO 138  I = 1, N, 1
               A = PI * I/N
                 DO 238 K = 1, J, 1
                      B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*(SOMEGA + B))
           XNGLE5 = SIN(2.*A)
```

```
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

C Note my 'APAS' here as above is in CGS.

           TNS=APAS*(TNS1 - TNS2 + TNS3)

C  CONDITIONAL LOGIC FOR STATES IN TIME-CALCULATION LOOP
C  THIS LOOP INCREMENTS TIME WITH THE NEW FREQUENCY, AND
C  CALCULATES THE VALUE OF <STATE> FOR EACH TIME INTERVAL

C              Calculation for <eqn A>

            TIMRE=0.25*PI*SIN(A)*TNORM*
     &(COS(1.5*TNS*TIME*PI)+COS((0.5*TNS + JJJ)*PI*TIME))
            TIMERS = TIMRE+TIMERS

238          CONTINUE
138        CONTINUE



C ----------------- CALC FOR EQN B --------------------------

          ELSE IF (STATE.EQ.'B') THEN

          J = N

           DO 139  I = 1, N, 1
               A = PI * I/N
                 DO 239 K = 1, J, 1
                      B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*(SOMEGA + B))
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

           TNS=APAS*(TNS1 - TNS2 + TNS3)

            TIMRE=0.25*PI*SIN(A)*TNORM*SIGN*
     &(COS(3/2*TNS*PI*TIME)-COS((0.5*TNS+JJJ)*PI*TIME))
            TIMERS=TIMRE+TIMERS

239          CONTINUE
139        CONTINUE



C--------------- CALC FOR EQN C ---------------------

         ELSE IF (STATE.EQ.'C') THEN

          J = N

           DO 141  I = 1, N, 1
               A = PI * I/N
               DO 241 K = 1, J, 1
                     B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*SOMEGA + 2.*B)
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

           TNS=APAS*(TNS1 - TNS2 + TNS3)

C              Calculation for <eqn C>

            TIMRE=0.25*PI*SIN(A)*TNORM*
     &(1+COS((JJJ-TNS)*PI*TIME))

           TIMERS=TIMRE+TIMERS

241          CONTINUE
141         CONTINUE
```

```
C--------------- CALC FOR EQN D ---------------------

        ELSE IF (STATE.EQ.'D') THEN

         J = N

          DO 142  I = 1, N, 1
              A = PI * I/N
                DO 242 K = 1, J, 1
                     B = 2 * PI * K/J

          XNGLE2 = SIN(A)**2
          XNGLE3 = COS(2.*SOMEGA + 2.*B)
          XNGLE5 = SIN(2.*A)
          XNGLE6 = COS(SOMEGA+B)
          XNGLE8 = 3.*(COS(A))**2

          TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
          TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=APAS*(TNS1 - TNS2 + TNS3)


C              Calculation for <eqn D>

           TIMRE=0.25*PI*SIN(A)*TNORM*
     &(1-COS((JJJ-TNS)*PI*TIME))
           TIMERS = TIMERS+TIMRE

242         CONTINUE
142        CONTINUE



C--------------- CALC FOR EQN E ---------------------

        ELSE IF (STATE.EQ.'E') THEN

         J = N

          DO 143  I = 1, N, 1
              A = PI * I/N
                DO 243 K = 1, J, 1
                     B = 2 * PI * K/J

          XNGLE2 = SIN(A)**2
          XNGLE3 = COS(2.*SOMEGA + 2.*B)
          XNGLE5 = SIN(2.*A)
          XNGLE6 = COS(SOMEGA+B)
          XNGLE8 = 3.*(COS(A))**2

          TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
          TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

          TNS=APAS*(TNS1 - TNS2 + TNS3)


C              Calculation for <eqn E>

           TIMRE=0.25*PI*SIN(A)*TNORM*SIGN*
     &(SIN((0.5*TNS+JJJ)*PI*TIME) + SIN(1.5*TNS*PI*TIME))
           TIMERS=TIMRE+TIMERS

243         CONTINUE
143        CONTINUE



C--------------- CALC FOR EQN F  ---------------------

        ELSE IF (STATE.EQ.'F') THEN

         J = N

          DO 145  I = 1, N, 1
              A = PI * I/N
                DO 245 K = 1, J, 1
                     B = 2 * PI * K/J

          XNGLE2 = SIN(A)**2
          XNGLE3 = COS(2.*SOMEGA + 2.*B)
          XNGLE5 = SIN(2.*A)
          XNGLE6 = COS(SOMEGA+B)
          XNGLE8 = 3.*(COS(A))**2

          TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
          TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
          TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))
```

```
          TNS=APAS*(TNS1 - TNS2 + TNS3)


C               Calculation for <eqn F>

           TIMRE= 0.25*PI*SIN(A)*TNORM*SIGN*
     &(SIN((0.5*TNS+JJJ)*PI*TIME) - SIN(1.5*TNS*PI*TIME))
              TIMERS=TIMRE+TIMERS

245          CONTINUE
145        CONTINUE


C--------------- CALC FOR EQN G  ---------------------

         ELSE IF (STATE.EQ.'G') THEN

          J = N

           DO 147  I = 1, N, 1
              A = PI * I/N
                DO 247 K = 1, J, 1
                     B = 2 * PI * K/J

           XNGLE2 = SIN(A)**2
           XNGLE3 = COS(2.*SOMEGA + 2.*B)
           XNGLE5 = SIN(2.*A)
           XNGLE6 = COS(SOMEGA+B)
           XNGLE8 = 3.*(COS(A))**2

           TNS1 = ((3./4.) * XNGLE1 * XNGLE2 * XNGLE3)
           TNS2 = ((3./4.) * XNGLE4 * XNGLE5   * XNGLE6 )
           TNS3 =  ((1./4.) * (XNGLE7 -1) * (XNGLE8 -1))

           TNS=APAS*(TNS1 - TNS2 + TNS3)


C               Calculation for <eqn G>

       TIMRE=0.25*PI*SIN(A)*SIGN*TNORM*SIN((JJJ-TNS)*PI*TIME)
        TIMERS=TIMRE+TIMERS

247          CONTINUE
147        CONTINUE


         ENDIF
```

```
C Some useful write-tos to the output file.
        IF (OUTPUT.EQ.1) THEN
       WRITE (4,*)' RESULTS FOR: ', STATE
       WRITE (4,*)' --------------------------'
       WRITE (4,*)' '
       WRITE (4,*)' No. of steps in integration: ', N
       WRITE (4,*)' Gamma for homonuclear pair: ', GAMMA
       WRITE (4,*)' No. of data points reporting:', M
       WRITE (4,*)' Spinning Angle: ', ANGLE
        WRITE (4,*)' Stroboscopic Sample Time ', STIME
        WRITE (4,*)' Internuclear Distance (angstroms): ', R*1E8
       WRITE (4,*)' Spinning Speed: ' , OMEGAR
       WRITE (4,*)' ----------------'
        WRITE (4,*)' Evolution of EQN ', STATE , ' in time steps
of',
     &TSTEP*1E03, ' milliseconds'
       WRITE (4,*)' ----------------'

        ENDIF

C  Write to file. Re-initialize 'Result'.

       OUTPUT=0
            WRITE(4,*) DIPVAL,  TIMERS
           TIMRE=0.
            TIMERS=0.

        DIPVAL = DIPVAL + DIPSTP
        IF (DIPVAL.LT.DIPMAX) THEN
        GOTO 59
        ENDIF

        END
```

**PROGRAM RFDR**

```
C This program calculates the response of a two-spin
C system to the dipolar Hamiltonian (Hd) when one of
C he spins (the I spin) is perturbed. It can also
C include isotropic J as well as D. It then graphs
C the FID response of spin S as well as spin I to two
C seperate files so you can plot them. This also
C nvolves itself with FFT calculations. Comments provided
C for more clue.
C The average Hamiltonian employed is that of
C the RFDR experiment (see Biblio, ref.
```

```
      Implicit double precision (a-h, o-z)
      INTEGER *2 FL
      REAL TSTEP, SIGN
      DOUBLE PRECISION PI,HBAR,APAS,CST1,CST2
      DOUBLE PRECISION XMAX,TNG,ADJ,CST3, GAL
      CHARACTER *30 FID, FFTOUT
      CHARACTER *1 STATE, SIGNI, FFTYN
      DIMENSION  RES1(10000),  RESULT1(10000), CONFIG(10000)
      DIMENSION AT(131074), BT(65536), CT(65536), DT(65536)
      DATA PI /3.1415 92653 58979 324/
C     Planck's constant in cgs (erg sec rad(-1))
      DATA HBAR /1.0545887E-27/

      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
      PRINT *, ''
5     PRINT *, 'Calculations are in cgs.'
      PRINT *, ''
      PRINT *,''

      PRINT *, 'Input the eqn you want to calc (A,B,C,D,E)'
      READ (*,'(A)') STATE

      IF ((STATE.EQ.'A').OR.(STATE.EQ.'B').OR.(STATE.EQ.'C').OR.
     &(STATE.EQ.'D').OR.(STATE.EQ.'E')) THEN
      STATE=STATE
      ELSE
      PRINT *, 'You must enter a state of A, B, C, D, or E.
     &Please try again.'
      GOTO 5
      ENDIF

      PRINT *, 'Is this equation a negative? -A, -B, etc? (Y or
     N)'
      READ (*,'(A)') SIGNI

      IF ((SIGNI.EQ.'Y').OR.(SIGNI.EQ.'y')) THEN
      SIGN= -1.0
      ELSE
         SIGN=1.0
      ENDIF
       PRINT *, 'What is the name for the data file?'
       READ (*, '(A)') FID
       OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')

C  Parameter Input

       PRINT *, 'Parameters'
       PRINT *, ''
       PRINT *, 'Ratio of chemical shift difference to rotor
     frequency?'
       READ *, XMAX
       PRINT *, 'Value of Gamma for homonuclear pair?'
       PRINT *, '(C13 = 6727.0, H = 26750.0, in rad sec(-1) G(-
     1))'
       READ  *, GAMMA
       PRINT *, 'Distance between nuclei? (in Angstroms)'
       READ *, R
       PRINT *, 'Number of steps in summation per pass?'
       READ *, N
       PRINT *, 'Do you want an FFT calculated?'
       READ *, FFTYN

       IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN
        PRINT *, 'Number of steps in time array? Power of 2, up
     to 4096'
        READ *, M
        PRINT *, 'Name of FFT output file?'
        READ *, FFTOUT
       ELSE
        PRINT *, 'Number of steps in time array? (Even # up to
     10,000)'
        READ *, M

       ENDIF


C File Opening Units

C  UNIT CONVERSIONS AND ETC (cgs system)

C  R is the internuclear distance converted into cm.

       R = R * 1.E-8

C  TNORM is the normalization constant so when we add up
     summation
C  in the do loops, it normalizes to a max magnitude (amplitude)
     of 1.
C It's squared because the double sum goes by N^2.
```

```
       TNORM = 1./(N**2)

C  APAS is the constant in front of the dipolar hamiltonian...
C  gamma1*gamma2*hbar/R^3...since gamma1=gamma2, it's just
squared.

C  Note: in powders, there is no factor of 1/2 or any other
multiplier
C  with this simple dipolar constant. DIP is in rad/sec, APAS
C  is the modification due to the factor of 1/2 in the product
C  operator

      DIP=-(GAMMA**2)*HBAR*(1./R**3)
      APAS=DIP/2

C TEST determines the measure of time scale (see below).

       TEST=ABS(APAS)

       PRINT *, 'Value of spinning angle = ', ANGLE, 'radians.'
       PRINT *, 'Value of APAS =', APAS
       PRINT *, 'Distance in cm: ',R
       PRINT *, 'The gamma pair: ' , GAMMA

C      TIMESTEP CALCULATION using TEST (these seem to work well)

       IF (TEST.LT.1000.) THEN
              TSTEP=.0001
       ELSE IF (TEST.LT.5000.) THEN
              TSTEP=.00001
       ELSE IF (TEST.LT.10000.) THEN
              TSTEP=.00001
       ELSE IF (TEST.LT.20000.) THEN
              TSTEP=.00001
       ELSE IF (TEST.LT.50000.) THEN
              TSTEP=.00001
       ELSE
              TSTEP=.00001
       ENDIF
       PRINT *, 'Timesteps selected: ', TSTEP, ' seconds.'
       PRINT *, 'Your total data set will span ', TSTEP*M*1E03,
     &' milliseconds.'

C Pieces of the spatial part of the Hamiltonian which determines
the
C frequencies we will sum over.
C I broke it up to make spot-checks easier and to cut down
calculation
C in the loops.
C This is a simple Riemann sum. Higher the sum amount, more
accurate the
C FID-type powder spectra. The weighting term "SIN (THETA)" has
been included.

C REMINDER:
C The angle A is the angle THETA in the lab frame. The angle B is
PHI.
C ANGLE is the angle of the goniometer.

C XMAX is the maximum response of chemical shift difference per
rotor rotation
C from the SEDRA function for d. See figure in paper. Adding
together m=1
C and m=2 components got a maxmimum delta-shift/vr at 1.03362
which I
C derived in mathematica, with a bunch of assumptions since the
paper isn't
C clear at all. Like, we assume to put gamma^2/r^3 etc with the
wigner terms,
C right?

        TNG = (XMAX)**2.0
       CST1= -0.5*SQRT(3./2.)*XMAX/(1.-TNG)
        CST2= -0.5*SQRT(3./2.)*XMAX/(4.-TNG)
       CST3= 2.0/PI * SIN(XMAX*PI)

89       PRINT *, 'Now calculating FID for Equation ', STATE, '.'


C ---------------------- CALCULATIONS FOR EQN C -----------------

             IF (STATE.EQ.'C') THEN

C  Calculational doloop. One nested for theta, the other for phi.
C  A is theta, B is phi. The idea is to generate one frequency
C  over a range of frequencies from theta=0...pi and phi=0...2*pi
C  and then for each possible frequency (for each crystallite
orientation
C that goes by N^2), calculate ALL the time domain.

C  Then, you add each time point for each frequency into the
array
```

```
C  'Result1' and 'Result2' where the arrays are indexed by the
time
C   points.   This sum of all possible frequencies will result in
a kind
C  of FID-like response which is due to the multiple dipolar
frequencies.

C Here, the J=N  assures that the  J LOOP is the same as N for
PHI (B) as
C for THETA (A), seperate for tweaking.

          J = N

           DO 138  I = 1, N, 1
               A = PI * I/N
                 DO 238 K = 1, J, 1
                      B = 2 * PI * K/J
C Sum elements for the dipolar term in the SEDRA paper (d = sum
over m=1 and m=2)
C These are d12(beta) for m=1 and m=2


            TWK1 =SIN(2*A)*COS(B)*CST1
            TWK2 = SIN(A)**2*COS(2*B)*CST2
            TNS=APAS*CST3*(TWK2+TWK1)
            FNT=0.5*PI*SIN(A)*TNORM

C  CONDITIONAL LOGIC FOR STATES IN TIME-CALCULATION LOOP
C  THIS LOOP INCREMENTS TIME WITH THE NEW FREQUENCY, AND
C  CALCULATES THE VALUE OF <STATE> FOR EACH TIME INTERVAL

C               Calculation for <eqn C>
C               (Transfer from Iz to Iz)

       DO 11 NTIME = 0, M, 1
        DELTAT = NTIME * TSTEP
       RES1(NTIME)=FNT*COS(TNS*DELTAT)**2
         RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
11       CONTINUE

238          CONTINUE
138        CONTINUE



C ----------------- CALC FOR EQN E --------------------------
C      (Transfer from Iz to IxSy (-E) and IySx (E) )
```

```
          ELSE IF (STATE.EQ.'E') THEN

          J = N

           DO 139  I = 1, N, 1
              A = PI * I/N
                DO 239 K = 1, J, 1
                     B = 2 * PI * K/J


            TWK1 =SIN(2*A)*COS(B)*CST1
            TWK2 = SIN(A)**2 * COS(2*B)*CST2
            TNS=APAS*CST3*(TWK2+TWK1)
            FNT=0.5*PI*SIN(A)*TNORM

C               Loop for <eqn E>

         DO 12 NTIME = 0, M, 1
            DELTAT = NTIME * TSTEP
            RES1(NTIME)=FNT*0.5*SIN(2*TNS*DELTAT)
            RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
12            CONTINUE

239           CONTINUE
139         CONTINUE



C---------------- CALC FOR EQN D ---------------------
C           (Transfer from Iz to Sz)

          ELSE IF (STATE.EQ.'D') THEN

           J = N

            DO 141  I = 1, N, 1
               A = PI * I/N
               DO 241 K = 1, J, 1
                     B = 2 * PI * K/J


            TWK1 = SIN(2*A)*COS(B)*CST1
            TWK2 = SIN(A)**2 * COS(2*B)*CST2
```

```
            TNS=APAS*CST3*(TWK2+TWK1)
            FNT=0.5*PI*SIN(A)*TNORM

C               Calculation for <eqn D>

          DO 14 NTIME = 0, M, 1
             DELTAT = NTIME * TSTEP
             RES1(NTIME)=FNT*SIN(TNS*DELTAT)**2
             RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
14        CONTINUE

241           CONTINUE
141          CONTINUE



C--------------- CALC FOR EQN A---------------------
C (Transfer from  Ix to Ix, Sx to Sx, etc).

          ELSE IF (STATE.EQ.'A') THEN

           J = N
            DO 142  I = 1, N, 1
                A = PI * I/N
                  DO 242 K = 1, J, 1
                       B = 2. * PI * K/J


             TWK1 = SIN(2*A)*COS(B)*CST1*CST3
             TWK2 = SIN(A)**2 * COS(2*B)*CST2*CST3
             TNS=APAS*(TWK2+TWK1)
             FNT=0.5*PI*SIN(A)*TNORM

C               Loop Calculation for <eqn A>

          DO 15 NTIME = 0, M, 1
             DELTAT = NTIME * TSTEP
             RES1(NTIME)=FNT*SIGN*COS(TNS*DELTAT)
             RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
15        CONTINUE

242           CONTINUE
142          CONTINUE
```

```
C--------------- CALC FOR EQN B ---------------------
C (transfer from Sy to IxSz, usually 0 like EQN B)

          ELSE IF (STATE.EQ.'B') THEN

           J = N

            DO 143  I = 1, N, 1
                A = PI * I/N
                  DO 243 K = 1, J, 1
                       B = 2 * PI * K/J

             TWK1 = SIN(2*A)*COS(B)*CST1
             TWK2 = SIN(A)**2 * COS(2*B)*CST2
             TNS=APAS*CST3*(TWK2+TWK1)
             FNT=0.5*PI*SIN(A)*TNORM

C               Calculation for <eqn B>

          DO 16 NTIME = 0, M, 1
             DELTAT = NTIME * TSTEP
             RES1(NTIME)=FNT*SIGN*SIN(TNS*DELTAT)
             RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
16        CONTINUE
243           CONTINUE
143          CONTINUE

          ENDIF

C Some useful write-tos to the output file.

         WRITE (4,*)' RESULTS FOR: ', STATE
         WRITE (4,*)' --------------------------'
         WRITE (4,*)' '
         WRITE (4,*)' No. of crystallite orientations: ', N**2
         WRITE (4,*)' Gamma for homonuclear pair: ', GAMMA
         WRITE (4,*)' No. of data points reporting:', M
         WRITE (4,*)' Ratio of del-CS/vr: ',    XMAX
         WRITE (4,*)' Internuclear Distance (angstroms): ', R*1E8
         WRITE (4,*)' Value of D: ', APAS
         WRITE (4,*)' ----------------'
         WRITE (4,*)' Evolution of EQN ', STATE , ' in time steps
of',
       &TSTEP*1E03, ' milliseconds'
         WRITE (4,*)' ----------------'

         PRINT *, 'Writing to FID file...'
```

```
      DO 111 L=0, M, 1
           WRITE(4,*)  L*1E03*TSTEP,  '   ' , RESULT1(L)
111      CONTINUE

         WRITE(4,*)'-9999'

9999   IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN

        DO 107 I=1, M, 1
        CONFIG(I)=RESULT1(I)
107     CONTINUE

        DO 109  I=1, M, 1
        WRITE (3,*) I*1E03*TSTEP, '    ', CONFIG(I)
109     CONTINUE

      CALL FFT(CONFIG,M,FFTOUT,TSTEP,STATE,GAMMA,
     &N,APAS)
       ENDIF
       END

      SUBROUTINE FFT (AT,M,FFTOUT,TSTEP,STATE,GAMMA,
     &N,APAS)
      REAL TSTEP
      DOUBLE PRECISION PI,HBAR,APAS,TAU1,TAU2,RATIO,GAMMA,GNUR
      CHARACTER *1 STATE, SIGNI, FFTYN
      INTEGER I, J
      DOUBLE PRECISION AT(M+2),BT(M/2+1),CT(M/2+1),DT(M/2+1)
      CHARACTER * 30 FFTOUT
      DATA PI /3.1415 92653 58979 324/
      OPEN(5, FILE=FFTOUT)
      CLOSE(4)
      CALL SFFT2(M/2,AT,1)
      CALL SFS(M,AT,1)
      PRINT *, 'Writing to FFT file...'
      WRITE (5,*), 'FFT CALCULATION FOR EQN ', STATE
       WRITE (5,*)' --------------------------'
       WRITE (5,*)' '
       WRITE (5,*)' No. of steps in integration: ', N
       WRITE (5,*)' Gamma for homonuclear pair: ', GAMMA
       WRITE (5,*)' No. of data points reporting:', M
       WRITE (5,*)' Internuclear Distance (angstroms): ', R*1E8
       WRITE (5,*)' RATIO inner evolution to rot period:', RATIO
       WRITE (5,*)' TAU1:' , TAU1, 'TAU2:' , TAU2
       WRITE (5,*)' Spinning Speed: ' , GNUR ,  ' Hz'
       WRITE (5,*)' Value of D: ', APAS
       WRITE (5,*)' ----------------'
        WRITE (5,*)' FFT of EQN ', STATE , ' in freq steps of',
     &1/(M*1E03*TSTEP), ' milliseconds'
        WRITE (5,*)' ----------------'

C Removing that first point
C CREATING THE DIPOLAR POWDER PATTERN: PRINT OUT THE LEFT SIDE BY
REVERSING
C THE SOLUTION FROM LEFT TO RIGHT, THEN PLOT THE FFT FOR THE
RIGHT HAND
C SIDE OF THE PLOT.

      DO 200 J=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(J)=AT(M-(2*J-1))
200   CONTINUE

      DO 201 J=1, M/2
      DT(J)=BT(J-1)
      WRITE(5,*), (J-(M/2+1))/(M*TSTEP), DT(J)*M
201   CONTINUE

      DO 202 I=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(I)=AT(2*I-1)
      CT(I)=AT(2*I)
      WRITE(88,*),CT(I)
202   CONTINUE
      BT(M/2 +1)=0.
      DO  203 I=1, M/2
      DT(I)=BT(I+1)
      WRITE(5,*), I/(M*TSTEP), DT(I)*M
203   CONTINUE

C Finish off with a tag for the data file (for perl script
purposes)

      WRITE(5,*), '-9999'
      RETURN
      END

C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFFT2 (N,Z,SIGN)
```

```
      double precision Z(N)
      INTEGER N, SIGN
      INTEGER JBH, JBL, KR, LOG2
      INTEGER BLK, KK, K0, K1, K2, K3, N2, SPAN
      DOUBLE PRECISION CE, C1, C2, C3, SE, S1, S2, S3, PIOV2,
     THETA
      DOUBLE PRECISION  X1, X2, X3, Y1, Y2, Y3
      DATA PIOV2/1.5707 96326 79489 662/
      N2 = 1
      LOG2 = 0
 8100    LOG2 = LOG2 + 1
         N2 = N2 + N2
         IF ( N2 .LT. N )  GO TO 8100
      IF ( N2 .GT. N ) RETURN
      N2 = N2 + N2 - 1
      IF ( SIGN .GE. 0 ) GO TO 8210
       DO 8200 KK = 1, N2, 2
 8200      Z(KK+1) = -Z(KK+1)
 8210 CONTINUE
      JBL = 2
      JBH = N
 8300 IF ( JBH .LE. JBL ) GO TO 8340
         SPAN = JBL + JBL
         KR = 1
 8310        KK = KR + JBL
             KR = KR + JBH
             BLK = KR - JBL
             DO 8330 K1 = KK, BLK, SPAN
                 K3 = K1 + JBL - 2
                 DO 8320 K0 = K1, K3, 2
                     X1 = Z(K0)
                     Z(K0) = Z(KR)
                     Z(KR) = X1
                     Y1 = Z(K0+1)
                     Z(K0+1) = Z(KR+1)
                     Z(KR+1) = Y1
 8320                KR = KR + 2
                 KR = KR + JBL
 8330        CONTINUE
             IF ( KR .LT. N2 ) GO TO 8310
         JBH = JBH / 2
         JBL = JBL + JBL
         GO TO 8300
 8340 CONTINUE
      IF ( MOD(LOG2,2) .NE. 0 ) GO TO 8400
       THETA = PIOV2
       SPAN = 2
       GO TO 8420
 8400 CONTINUE
       DO 8410 K0 = 1, N2, 4
           K1 = K0 + 2
           X1 = Z(K0)
           Z(K0) = X1 + Z(K1)
           Z(K1) = X1 - Z(K1)
           Y1 = Z(K0+1)
           Z(K0+1) = Y1 + Z(K1+1)
 8410      Z(K1+1) = Y1 - Z(K1+1)
       THETA = PIOV2 + PIOV2
       SPAN = 4
 8420 CONTINUE
 8500 IF ( SPAN .GE. N2 ) GO TO 8560
         BLK = SPAN * 4
         IF ( SPAN .EQ. 2 ) GO TO 8505
           THETA = .25 * THETA
           SE = SIN(THETA)
           CE = COS(THETA)
           C1 = 1.0
           S1 = 0.0
 8505    CONTINUE
         DO 8550 KK = 1, SPAN, 2
             IF ( KK .EQ. 1 ) GO TO 8510
               X1 = C1 * CE - S1 * SE
               Y1 = C1 * SE + S1 * CE
               X2 = .5 * ( 3. - ( X1 * X1 + Y1 * Y1 ))
               C1 = X2 * X1
               S1 = X2 * Y1
               C2 = C1 * C1 - S1 * S1
               S2 = C1 * S1 + C1 * S1
               C3 = C2 * C1 - S2 * S1
               S3 = C2 * S1 + S2 * C1
 8510        CONTINUE
             DO 8540 K0 = KK, N2, BLK
                K1 = K0 + SPAN
                K2 = K1 + SPAN
                K3 = K2 + SPAN
                IF ( KK .GT. 1 ) GO TO 8520
                  X1 = Z(K1)
                  Y1 = Z(K1+1)
                  X2 = Z(K2)
                  Y2 = Z(K2+1)
                  X3 = Z(K3)
                  Y3 = Z(K3+1)
                  GO TO 8530
 8520             CONTINUE
```

```
                X1 = Z(K1) * C2 - Z(K1+1) * S2
                Y1 = Z(K1) * S2 + Z(K1+1) * C2
                X2 = Z(K2) * C1 - Z(K2+1) * S1
                Y2 = Z(K2) * S1 + Z(K2+1) * C1
                X3 = Z(K3) * C3 - Z(K3+1) * S3
                Y3 = Z(K3) * S3 + Z(K3+1) * C3
 8530           CONTINUE
                Z(K3) = Z(K0) - X1 + Y2 - Y3
                Z(K2) = Z(K0) + X1 - X2 - X3
                Z(K1) = Z(K0) - X1 - Y2 + Y3
                Z(K0) = Z(K0) + X1 + X2 + X3
                Z(K3+1) = Z(K0+1) - Y1 - X2 + X3
                Z(K2+1) = Z(K0+1) + Y1 - Y2 - Y3
                Z(K1+1) = Z(K0+1) - Y1 + X2 - X3
                Z(K0+1) = Z(K0+1) + Y1 + Y2 + Y3
 8540         CONTINUE
 8550     CONTINUE
          SPAN = BLK
          GO TO 8500
 8560 CONTINUE
      IF ( SIGN .GE. 0 ) RETURN
       DO 8600 KK = 1, N2, 2
 8600       Z(KK+1) = -Z(KK+1)
      RETURN
      END

C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS
      SUBROUTINE SFS (N, Z, SIGN)
      DOUBLE PRECISION Z(n)
      INTEGER N, SIGN
      INTEGER K0, K1, ND2
      DOUBLE PRECISION PI, CE, C1, SE, S1, SF, X1, X2, Y1, Y2
      DATA PI /3.1415 92653 58979 324/
      ND2 = IABS(N) / 2
      IF (ND2 + ND2 .NE. N) RETURN
      SF = 1. / FLOAT(ND2)
      X1 = PI * SF
      IF (SIGN .GE. 0) GO TO 8100
        SF = .5
        S1 = 1.
        Z(2) = SF * (Z(1) - Z(N+1))
        Z(1) = SF * (Z(1) + Z(N+1))
        X1 = -X1
        GO TO 8110
 8100 CONTINUE
        Z(N+1) = SF * (Z(1) - Z(2))
        Z(N+2) = 0.0
        Z(1) = SF * (Z(1) + Z(2))
        Z(2) = 0.0
        IF (MOD(ND2,2) .NE. 0) GO TO 8105
          Z(ND2+1) = SF * Z(ND2+1)
          Z(ND2+2) = SF * Z(ND2+2)
 8105   CONTINUE
        SF = .5 * SF
        S1 = -1.
 8110 CONTINUE
      C1 = 0.0
      CE = COS(X1)
      SE = SIN(X1)
      K1 = N - 1
      K0 = 3
 8200 IF (K0 .GE. K1) RETURN
        X1 = CE * C1 - SE * S1
        Y1 = CE * S1 + SE * C1
        X2 = .5 * (3. - (X1 * X1 + Y1 * Y1))
        C1 = X1 * X2
        S1 = Y1 * X2
        X1 = Z(K0) - Z(K1)
        Y1 = Z(K0+1) + Z(K1+1)
        X2 = C1 * X1 - S1 * Y1
        Y2 = C1 * Y1 + S1 * X1
        X1 = Z(K0) + Z(K1)
        Y1 = Z(K0+1) - Z(K1+1)
        Z(K0) = SF * (X1 + X2)
        Z(K0+1) = SF * (Y1 + Y2)
        Z(K1) = SF * (X1 - X2)
        Z(K1+1) = SF * (Y2 -Y1)
        K1 = K1 - 2
        K0 = K0 + 2
        GO TO 8200
      END
```

**PROGRAM IS.F**

```
      IMPLICIT DOUBLE PRECISION (a-h, o-z)
      DOUBLE PRECISION JJJ1, JJJ, NUCGAM
         CHARACTER *30 FID, INPT, FFTOUT
      CHARACTER *1 STATE, SIGNI, FFTYN
         CHARACTER *2 STATID

         DIMENSION RES1(100000), RESULT1(100000), CONFIG(100000)
         DIMENSION AT(131138), BT(131138), CT(131138),
DT(131138)
      INTEGER N,M,I,IL,J,K
```

```
      COMPLEX wip(-2:2),wic(-2:2),wir(-2:2),wil(-2:2)
      COMPLEX wsp(-2:2),wsc(-2:2),wsr(-2:2),wsl(-2:2)
      COMPLEX wisp(-2:2),wisc(-2:2), wisr(-2:2), wisl(-2:2)
      COMPLEX D2(-2:2,-2:2)

      DATA PI /3.1415 92653 58979 324/
      DATA HBAR /1.0545887E-27/
C  proton gyromagnetic ratio  = 26750.0, in rad sec(-1) G(-1))
         DATA HGAM /26750.0/

5     PRINT *, 'Calculations are in CGS.'
      PRINT *, ''
      PRINT *, ''
      PRINT *, 'Input the intial state.'
          PRINT *, 'Type A for Ix, and B for Iz.'
      PRINT *, ''
          READ (*, '(A)') STATE
          IF ((STATE.EQ.'A').OR.(STATE.EQ.'B')) THEN
          STATE=STATE
          ELSE
      PRINT *, 'Enter a state A or B only. Try again.'
          GOTO 5
          ENDIF

      PRINT *, 'Name of Input File'
          READ (*, '(A)') INPT
          PRINT *, 'What is the name for the output file?'
          READ (*, '(A)') FID
      PRINT *, 'Do you want an FFT calculated?'
      READ *, FFTYN

      IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN
            PRINT *, "Make sure your input file timesteps are"
                PRINT *, "an integer power of 2."
                PRINT *, "----------"
            PRINT *, 'Name of FFT output file?'
            READ *, FFTOUT
          ELSE
            PRINT *, "Calculating..."
          ENDIF

          OPEN (UNIT=4, FILE=FID, STATUS='UNKNOWN')
          OPEN (UNIT=10,FILE=INPT,STATUS='UNKNOWN')

          READ(10,*) HNU0,NUCGAM,R,JJJ1
          PRINT *, HNU0, NUCGAM, R, JJJ1, CGAM

C     Chemical shifts in ppm, for spins I (csi) and spin S (css)

          READ(10,*) csix, csiy, csiz
          PRINT *, csix, csiy, csiz
          READ(10,*) cssx, cssy, cssz
          PRINT *, cssx, cssy, cssz
          READ(10,*) N, TIME, M
          PRINT *, N, TIME, M
          READ(10,*) ai_p2c, bi_p2c, gi_p2c
          PRINT *, ai_p2c, bi_p2c, gi_p2c
          READ(10,*) as_p2c, bs_p2c, gs_p2c
          PRINT *, as_p2c, bs_p2c, gs_p2c
          READ(10,*) a_c2r, b_c2r, g_c2r
          PRINT *, a_c2r, b_c2r, g_c2r

      rad=PI/180.
          vnu=HNU0 * 2.*PI * NUCGAM/HGAM
C Change ppm to rad/sec
          csix=csix*vnu
          cssx=cssx*vnu
          cssy=cssy*vnu
          cssz=cssz*vnu
          csiy=csiy*vnu
          csiz=csiz*vnu

          CSIi=1./3.*(csiy+csix+csiz)
      PRINT *, "CSIi = " , CSIi

          CSSi=1./3.*(cssy+cssx+cssz)
          PRINT *, "CSSi =" , CSSi

          ai_p2c=ai_p2c*rad
          bi_p2c=bi_p2c*rad
          gi_p2c=gi_p2c*rad

          as_p2c=as_p2c*rad
          bs_p2c=bs_p2c*rad
          gs_p2c=gs_p2c*rad

          ais_p2c=ais_p2c*rad
          bis_p2c=bis_p2c*rad
          gis_p2c=gis_p2c*rad

      a_c2r=a_c2r*rad
          b_c2r=b_c2r*rad
          g_c2r=g_c2r*rad
```

```
      ANISOI = csiz - csIi
          ANISOS = cssz - cSSi

          DIFFI=csix-csiy
          DIFFS=cssx-cssy

          etai=DIFFI/ANISOI
          etas=DIFFS/ANISOS
          R= R * 1.E-8
      DIP=-(NUCGAM**2)*HBAR*(1./R**3)

          PRINT *, ANISOI, ANISOS, DIFFI, DIFFS, etai, etas, R,
DIP
          JJJ=JJJ1*2.*PI
          TNORM3 = 1./(N**3)
          TNORM2=1./(N**2)
          TIME=TIME*1E-03
      TSTEP = TIME/M

          wip(0) = anisoi
          wip(2) = wip(0)*(-etai*sqrt(1./6.))
          wip(-2)= wip(2)
          wip(-1)= 0.
          wip(1) = 0.
          PRINT *, "wip = ", wip

          wsp(0) = anisos
          wsp(2) = wsp(0)*(-etas*sqrt(1./6.))
          wsp(-2)= wsp(2)
          wsp(-1)= 0.
          wsp(1) = 0.
          PRINT *, "wsp =", wsp

          wisp(0)= DIP
          wisp(2) = 0.
          wisp(-2)= 0.
          wisp(-1)= 0.
          wisp(1) = 0.

          wisc(0)=DIP
          wisc(2)=0.
          wisc(-2)=0.
          wisc(-1)=0.
          wisc(1)=0.

          DO 10 IG = 1, 5, 1
               DO 11 IT = 1, 6, 1
                  DIT=IT-3
                  DIG=IG-3
                  D2(DIT,DIG)=0.
11        CONTINUE
10        CONTINUE


          Call WIGNER2(D2, ai_p2c, bi_p2c, gi_p2c)
             Call c_vec_mat_mul(wip,D2(-2,-2),wic,-2,2,-2,2)
             PRINT *, "wic =", wic

          Call WIGNER2(D2,as_p2c,bs_p2c,gs_p2c)
             CALL c_vec_mat_mul(wsp,D2(-2,-2),wsc,-2,2,-2,2)
             PRINT *, "wsc =" , wsc

C         Now to rotor frame, convert from crystal frame.

             DO 510 IG = 1, 5, 1
               DO 511 IT = 1, 6, 1
                  DIT=IT-3
                  DIG=IG-3
                  D2(DIT,DIG)=0.
511        CONTINUE
510        CONTINUE

             CALL WIGNER2(D2,a_c2r,b_c2r,g_c2r)

             call c_vec_mat_mul(wsc,D2(-2,-2),wsr,-2,2,-2,2)
             PRINT *, "wsr =", wsr

             call c_vec_mat_mul(wic ,D2(-2,-2),wir,-2,2,-2,2)
             PRINT *, "wir =", wir

             call c_vec_mat_mul(wisc,D2(-2,-2),wisr,-2,2,-2,2)
             PRINT *, "wisr =", wisr



C------------------ Loop for Ix -------------------------------

          IF (STATE.eq.'A') THEN

             STATID='Ix'

             DO 138 I=1, N, 1
                BET=PI*I/N
```

```
C       DO 238 K=1, N, 1
C                     GAM=2.*PI*K/N
             GAM=0.0
                          DO 338 IL=1,N,1
                ALPH=2.*PI*IL/N

           CALL WIGNER2(D2,ALPH,BET,GAM)

           call c_vec_mat_mul(wir,D2(-2,-2),wil,-2,2,-2,2)

           call c_vec_mat_mul(wsr,D2(-2,-2),wsl,-2,2,-2,2)
           call c_vec_mat_mul(wisr,D2(-2,-2),wisl, -2,2,-2,2)

           wil(0)=(wil(0)+csIi)
                  wsl(0)=(wsl(0)+csSi)
           wisl(0)=wisl(0)

           omegai=real(wil(0))
           omegas=real(wsl(0))

                  SUMIS=(0.5*((omegai)+(omegas)))
           DIFFIS=(omegai-(omegas))
                  diptns=real(wisl(0))
                  ATERM=(JJJ+2.*DIPTNS)
                  BTERM=(JJJ-DIPTNS)
                  RTERM=SQRT(DIFFIS**2 + BTERM**2)

                  DO 38 NTIME=0,M,1
                  DELTAT = NTIME * TSTEP

                  RES1(NTIME)=TNORM2*Cos(ATERM*DELTAT)*
     &( COS(BTERM*DELTAT*0.5)*COS(SUMIS*DELTAT) +
     & (DIFFIS/RTERM)*SIN(BTERM*DELTAT*0.5)*SIN(SUMIS*DELTAT))
             RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)
38         CONTINUE
338        CONTINUE
C238        CONTINUE
138        CONTINUE

C ------------------ CALC FOR Iz -----------------------

      ELSE IF (STATE.eq.'B') THEN
      STATID='Iz'
         DO 148 I=1, N, 1
            BET = PI*I/N
               DO 248 K=1, N, 1
             GAM = 2.*PI*K/N
C                     DO 348 IL=1,N,1
C             ALPH=2.*PI*IL/N
              ALPH=0.0

           CALL WIGNER2(D2,ALPH,BET,GAM)
           call c_vec_mat_mul(wir(-2),D2(-2,-2),
     &wil,-2,2,-2,2)
           call c_vec_mat_mul(wsr(-2),D2(-2,-2),
     &wsl,-2,2,-2,2)
           call c_vec_mat_mul(wisr(-2),D2(-2,-2),
     &wisl,-2,2,-2,2)

           wil(0)=wil(0)+csIi
                  wsl(0)=wsl(0)+csSi
C                 wisl(0)=wisl(0)


                  DIFFIS=(real(wil(0)-wsl(0)))
                  diptns=(real(wisl(0)))

                  ATERM=JJJ+ 2.*DIPTNS
                  BTERM=JJJ - DIPTNS
                  BSQ= BTERM**2
           DIFFSQ=DIFFIS**2
                  RTERM = SQRT(DIFFSQ + BSQ)
                  RSQ=DIFSQ + BSQ

                  DO 48 NTIME=0,M,1

                  DELTAT = NTIME * TSTEP

                  RES1(NTIME)=TNORM2*SIN(BET)*0.25*PI*
     &((RSQ+DIFFSQ+BSQ*COS(BTERM*DELTAT))/(2.*RSQ))

                RESULT1(NTIME) = RESULT1(NTIME) + RES1(NTIME)

48         CONTINUE

C348        CONTINUE
248        CONTINUE
148        CONTINUE


          ENDIF

         WRITE (4,*) 'RESULTS FOR: ', STATID
```

```
      WRITE (4,*) '--------------------------'
      WRITE (4,*) 'H larmor B0, nuclei gamma, R Angs, J Hz '
         WRITE (4,*)  HNU0 ,  NUCGAM,  R ,  JJJ1
         WRITE (4,*) 'Chm shft ppm sigxx,sigyy,sigzz,CS PAS for
spin 1'
      WRITE (4,*)  csix/vnu, csiy/vnu, csiz/vnu
         WRITE (4,*) 'Chm shft ppm sigxx,sigyy,sigzz,CS PAS for
spin 2'
      WRITE (4,*) cssx/vnu, cssy/vnu, cssz/vnu
         WRITE (4,*) '# loop sum stps, sim time (ms), # plot
points'
         WRITE (4,*) N  ,   TIME   ,     M
      WRITE (4,*) 'Angles from PAS to crystal frame for spin I in
deg.'
         WRITE (4,*)  ai_p2c/rad , bi_p2c/rad , gi_p2c/rad
      WRITE (4,*) "Angles from PAS to crystal frame for spin 2 in
deg."
      WRITE (4,*) as_p2c/rad  , bs_p2c/rad , gs_p2c/rad
         WRITE (4,*) "Angles from crystal frame to rotor frame
in deg"
      WRITE (4,*)  a_c2r/rad  , b_c2r/rad  ,  g_c2r/rad
         WRITE (4,*) '(rotor->lab xform is done in riemann
loop)'
      WRITE (4,*) '----------------'


          PRINT *, 'Writing to FID file...'
       DO 111 L=0, M, 1
            WRITE(4,*)  L*1E03*TSTEP,  '   ' , RESULT1(L)

111      CONTINUE



9999   IF ((FFTYN.EQ.'Y').OR.(FFTYN.EQ.'y')) THEN

        DO 107 I=1, M, 1
                  CONFIG(I)=RESULT1(I)
107     CONTINUE

       CALL FFT(CONFIG,M,FFTOUT,TSTEP,STATED,NUCGAM,HNU0,
     &N, R, JJJ1, csix, csiy, csiz, cssx, cssy, cssz, vnu,
     &TIME, ai_p2c, bi_p2c, gi_p2c, rad, as_p2c, bs_p2c,
     &gs_p2c, a_c2r,b_c2r,g_c2r)

          ENDIF
```



```
          END



      SUBROUTINE WIGNER2(D2,ALPHA,BETA,GAMMA)
C
C     calculates second-rank Wigner matrix
C      this is the same as that given by spiess, assuming
c      first index = row, 2nd index = col.
C
      IMPLICIT DOUBLE PRECISION (a-h, o-z)
      SAVE
      COMPLEX D2(-2:2,-2:2)
      COMPLEX EM2AM2G,EM2AMG,EM2A,EM2APG,EM2AP2G
      COMPLEX EMAM2G,EMAMG,EMA,EMAPG,EMAP2G
      COMPLEX EM2G,EMG,EPG,EP2G
      COMPLEX EPAM2G,EPAMG,EPA,EPAPG,EPAP2G
      COMPLEX EP2AM2G,EP2AMG,EP2A,EP2APG,EP2AP2G
C
      COSB=COS(BETA)
      SINB=SIN(BETA)
      COS2B=COSB**2
      SIN2B=SINB**2
      CPLUS=(1.0+COSB)*0.5
      CMINUS=(1.0-COSB)*0.5
      R3BY8=SQRT(3.0/8.0)
      SINOF2B=SIN(2.0*BETA)
C
      EM2AM2G= CEXP(CMPLX(0.0,(-2.0*ALPHA-2.0*GAMMA)))
      EM2AMG= CEXP(CMPLX(0.0,-2.0*ALPHA-GAMMA))
      EM2A= CEXP(CMPLX(0.0,-2.0*ALPHA))
      EM2APG= CEXP(CMPLX(0.0,(-2.0*ALPHA+GAMMA)))
      EM2AP2G= CEXP(CMPLX(0.0,(-2.0*ALPHA+2.0*GAMMA)))
C
      EMAM2G= CEXP(CMPLX(0.0,(-ALPHA-2.0*GAMMA)))
      EMAMG= CEXP(CMPLX(0.0,(-ALPHA-GAMMA)))
      EMA= CEXP(CMPLX(0.0,-ALPHA))
      EMAPG= CEXP(CMPLX(0.0,(-ALPHA+GAMMA)))
      EMAP2G= CEXP(CMPLX(0.0,(-ALPHA+2.0*GAMMA)))
C
      EM2G= CEXP(CMPLX(0.0,(-2.0*GAMMA)))
      EMG= CEXP(CMPLX(0.0,(-GAMMA)))
      EPG= CONJG(EMG)
      EP2G= CONJG(EM2G)
C
      EPAM2G= CONJG(EMAP2G)
      EPAMG= CONJG(EMAPG)
```

```
      EPA= CONJG(EMA)
      EPAPG= CONJG(EMAMG)
      EPAP2G= CONJG(EMAM2G)
C
      EP2AM2G= CONJG(EM2AP2G)
      EP2AMG= CONJG(EM2APG)
      EP2A= CONJG(EM2A)
      EP2APG= CONJG(EM2AMG)
      EP2AP2G= CONJG(EM2AM2G)
C
C     assign elements (according to Spiess p.204)
C
      D2( 2, 2)=(CPLUS**2)*EM2AM2G
      D2( 2, 1)=-CPLUS*SINB*EM2AMG
      D2( 2, 0)=R3BY8*SIN2B*EM2A
      D2( 2,-1)=-CMINUS*SINB*EM2APG
      D2( 2,-2)=(CMINUS**2)*EM2AP2G
C
      D2( 1, 2)=CPLUS*SINB*EMAM2G
      D2( 1, 1)=(COS2B-CMINUS)*EMAMG
      D2( 1, 0)=-R3BY8*SINOF2B*EMA
      D2( 1,-1)=(CPLUS-COS2B)*EMAPG
      D2( 1,-2)=-CMINUS*SINB*EMAP2G
C
      D2(0, 2)=R3BY8*SIN2B*EM2G
      D2(0, 1)=R3BY8*SINOF2B*EMG
      D2(0, 0)=0.5*(3.0*COS2B-1.0)
      D2(0,-1)=-R3BY8*SINOF2B*EPG
      D2(0,-2)=R3BY8*SIN2B*EP2G
C
      D2(-1, 2)=CMINUS*SINB*EPAM2G
      D2(-1, 1)=(CPLUS-COS2B)*EPAMG
      D2(-1, 0)=R3BY8*SINOF2B*EPA
      D2(-1,-1)=(COS2B-CMINUS)*EPAPG
      D2(-1,-2)=-CPLUS*SINB*EPAP2G
C
      D2(-2, 2)=(CMINUS**2)*EP2AM2G
      D2(-2, 1)=CMINUS*SINB*EP2AMG
      D2(-2, 0)=R3BY8*SIN2B*EP2A
      D2(-2,-1)=CPLUS*SINB*EP2APG
      D2(-2,-2)=(CPLUS**2)*EP2AP2G
C
      RETURN
      END


         SUBROUTINE C_VEC_MAT_MUL(VEC1,MAT,VEC2,NLOW,
     & NHIGH,NLOW_MAIN,NHIGH_MAIN)
C  multiplies vector by square matrix; assumes dimensioning in
c  main program is the same in all dimensions.
         SAVE
         COMPLEX VEC1(NLOW_MAIN:NHIGH_MAIN),
     1        MAT(NLOW_MAIN:NHIGH_MAIN,NLOW_MAIN:NHIGH_MAIN),
     1        VEC2(NLOW_MAIN:NHIGH_MAIN)
C_____________________________________________
         DO ICOL=NLOW,NHIGH
         VEC2(ICOL)=0.0
            DO IROW=NLOW,NHIGH
          VEC2(ICOL)=VEC2(ICOL)+VEC1(IROW)
     1 *MAT(IROW,ICOL)
            END DO

         END DO

         RETURN
         END


         SUBROUTINE FFT (AT,M,FFTOUT,TSTEP,STATED,NUCGAM,HNU0,
     &N, R, JJJ1, csix, csiy, csiz, cssx, cssy, cssz, vnu,
     &TIME, ai_p2c, bi_p2c, gi_p2c, rad, as_p2c, bs_p2c,
     &gs_p2c, a_c2r, b_c2r, g_c2r)
             IMPLICIT DOUBLE PRECISION (a-h, o-z)
             DOUBLE PRECISION JJJ1, JJJ, NUCGAM
             CHARACTER *30 FFTOUT
             CHARACTER *1 FFTYN
             CHARACTER *2 STATID
         INTEGER I, J
         DOUBLE PRECISION AT(M+2),BT(M/2+1),CT(M/2+1),DT(M/2+1)
             SAVE
         DATA PI /3.1415 92653 58979 324/
         OPEN(5, FILE=FFTOUT)
         CLOSE(4)
         CALL SFFT2(M/2,AT,1)
         CALL SFS(M,AT,1)
         PRINT *, 'Writing to FFT file...'
         WRITE (5,*), 'FFT CALCULATION FOR EQN ', STATED
         WRITE (5,*) 'H larmor B0, nuclei gamma, R Angs, J Hz '
         WRITE (5,*)  HNU0 ,  NUCGAM,  R ,  JJJ1
         WRITE (5,*) 'Chm shft ppm sigxx,sigyy,sigzz,CS PAS for spin
I'
             WRITE (5,*)  csix/vnu, csiy/vnu, csiz/vnu
         WRITE (5,*) 'Chm shft ppm sigxx,sigyy,sigzz,CS PAS for spin
S'
```

```
         WRITE (5,*) cssx/vnu, cssy/vnu, cssz/vnu
       WRITE (5,*) '# loop sum stps, sim time (ms), # plot points'
         WRITE (5,*)  N ,   TIME,    M
       WRITE (5,*) 'Angles from PAS to crystal frame for spin I'
         WRITE (5,*)  ai_p2c/rad , bi_p2c/rad , gi_p2c/rad
       WRITE (5,*) "Angles from PAS to crystal frame for spin S"
         WRITE (5,*)  as_p2c/rad  , bs_p2c/rad , gs_p2c/rad
       WRITE (5,*) "Angles from crystal frame to rotor frame"
         WRITE (5,*)  a_c2r/rad  , b_c2r/rad  ,  g_c2r/rad
       WRITE (5,*) '(rotor->lab xform is done in riemann loop)'
       WRITE (5,*) 'FFT of EQN ', STATED , ' in freq steps of',
      &1/(M*1E03*TSTEP), ' milliseconds'
        WRITE (5,*)' ----------------'

C Removing that first point

      DO 201 J=1, M-2
      DT(J) = AT(J+2)
201   CONTINUE
      DT(M-1)=0.
      DT(M)=0.

C CREATING THE DIPOLAR POWDER PATTERN: PRINT OUT THE LEFT SIDE BY
REVERSING
C THE SOLUTION FROM LEFT TO RIGHT, THEN PLOT THE FFT FOR THE
RIGHT HAND
C SIDE OF THE PLOT.

      DO 200 J=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(J)=DT(M-(2*J-1))
      WRITE(5,*), (J-(M/2+1))/(M*TSTEP), BT(J)*M
200   CONTINUE

      DO 202 I=1, M/2
C     This assures the real (cosine) series by 2*I-1...odd
series.
C     Imaginary (sine series) would merely be 2*I...even series.
      BT(I)=DT(2*I-1)
      CT(I)=DT(2*I)
      WRITE(5,*), I/(M*TSTEP), BT(I)*M
      WRITE(88,*),CT(I)
202   CONTINUE

C Finish off with a tag for the data file (for perl script
purposes)

      WRITE(5,*), '-9999'
      RETURN
      END


C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFFT2 (N,Z,SIGN)
      double precision Z(N)
      INTEGER N, SIGN
      INTEGER JBH, JBL, KR, LOG2
      INTEGER BLK, KK, K0, K1, K2, K3, N2, SPAN
      DOUBLE PRECISION CE, C1, C2, C3, SE, S1, S2, S3, PIOV2,
THETA
      DOUBLE PRECISION  X1, X2, X3, Y1, Y2, Y3
      DATA PIOV2/1.5707 96326 79489 662/
      N2 = 1
      LOG2 = 0
 8100    LOG2 = LOG2 + 1
         N2 = N2 + N2
         IF ( N2 .LT. N )  GO TO 8100
      IF ( N2 .GT. N ) RETURN
      N2 = N2 + N2 - 1
      IF ( SIGN .GE. 0 ) GO TO 8210
       DO 8200 KK = 1, N2, 2
 8200      Z(KK+1) = -Z(KK+1)
 8210 CONTINUE
      JBL = 2
      JBH = N
 8300 IF ( JBH .LE. JBL ) GO TO 8340
         SPAN = JBL + JBL
         KR = 1
 8310        KK = KR + JBL
             KR = KR + JBH
             BLK = KR - JBL
             DO 8330 K1 = KK, BLK, SPAN
                K3 = K1 + JBL - 2
                DO 8320 K0 = K1, K3, 2
                    X1 = Z(K0)
                    Z(K0) = Z(KR)
                    Z(KR) = X1
                    Y1 = Z(K0+1)
                    Z(K0+1) = Z(KR+1)
                    Z(KR+1) = Y1
```

```
8320               KR = KR + 2
                KR = KR + JBL
8330         CONTINUE
             IF ( KR .LT. N2 ) GO TO 8310
           JBH = JBH / 2
           JBL = JBL + JBL
           GO TO 8300
8340  CONTINUE
      IF ( MOD(LOG2,2) .NE. 0 ) GO TO 8400
       THETA = PIOV2
       SPAN = 2
       GO TO 8420
8400  CONTINUE
       DO 8410 K0 = 1, N2, 4
           K1 = K0 + 2
           X1 = Z(K0)
           Z(K0) = X1 + Z(K1)
           Z(K1) = X1 - Z(K1)
           Y1 = Z(K0+1)
           Z(K0+1) = Y1 + Z(K1+1)
8410       Z(K1+1) = Y1 - Z(K1+1)
       THETA = PIOV2 + PIOV2
       SPAN = 4
8420  CONTINUE
8500  IF ( SPAN .GE. N2 ) GO TO 8560
         BLK = SPAN * 4
         IF ( SPAN .EQ. 2 ) GO TO 8505
           THETA = .25 * THETA
           SE = SIN(THETA)
           CE = COS(THETA)
           C1 = 1.0
           S1 = 0.0
8505     CONTINUE
         DO 8550 KK = 1, SPAN, 2
             IF ( KK .EQ. 1 ) GO TO 8510
               X1 = C1 * CE - S1 * SE
               Y1 = C1 * SE + S1 * CE
               X2 = .5 * ( 3. - ( X1 * X1 + Y1 * Y1 ))
               C1 = X2 * X1
               S1 = X2 * Y1
               C2 = C1 * C1 - S1 * S1
               S2 = C1 * S1 + C1 * S1
               C3 = C2 * C1 - S2 * S1
               S3 = C2 * S1 + S2 * C1
8510         CONTINUE
             DO 8540 K0 = KK, N2, BLK
                K1 = K0 + SPAN
                K2 = K1 + SPAN
                K3 = K2 + SPAN
                IF ( KK .GT. 1 ) GO TO 8520
                  X1 = Z(K1)
                  Y1 = Z(K1+1)
                  X2 = Z(K2)
                  Y2 = Z(K2+1)
                  X3 = Z(K3)
                  Y3 = Z(K3+1)
                  GO TO 8530
8520            CONTINUE
                  X1 = Z(K1) * C2 - Z(K1+1) * S2
                  Y1 = Z(K1) * S2 + Z(K1+1) * C2
                  X2 = Z(K2) * C1 - Z(K2+1) * S1
                  Y2 = Z(K2) * S1 + Z(K2+1) * C1
                  X3 = Z(K3) * C3 - Z(K3+1) * S3
                  Y3 = Z(K3) * S3 + Z(K3+1) * C3
8530            CONTINUE
                Z(K3) = Z(K0) - X1 + Y2 - Y3
                Z(K2) = Z(K0) + X1 - X2 - X3
                Z(K1) = Z(K0) - X1 - Y2 + Y3
                Z(K0) = Z(K0) + X1 + X2 + X3
                Z(K3+1) = Z(K0+1) - Y1 - X2 + X3
                Z(K2+1) = Z(K0+1) + Y1 - Y2 - Y3
                Z(K1+1) = Z(K0+1) - Y1 + X2 - X3
                Z(K0+1) = Z(K0+1) + Y1 + Y2 + Y3
8540          CONTINUE
8550      CONTINUE
          SPAN = BLK
          GO TO 8500
8560  CONTINUE
      IF ( SIGN .GE. 0 ) RETURN
       DO 8600 KK = 1, N2, 2
8600       Z(KK+1) = -Z(KK+1)
      RETURN
      END


C FAST FOURIER TRANSFORM SUBROUTINE TAKEN FROM MTS

      SUBROUTINE SFS (N, Z, SIGN)
      DOUBLE PRECISION Z(n)
      INTEGER N, SIGN
      INTEGER K0, K1, ND2
      DOUBLE PRECISION PI, CE, C1, SE, S1, SF, X1, X2, Y1, Y2
      DATA PI /3.1415 92653 58979 324/
      ND2 = IABS(N) / 2
```

```
      IF (ND2 + ND2 .NE. N) RETURN
      SF = 1. / FLOAT(ND2)
      X1 = PI * SF
      IF (SIGN .GE. 0) GO TO 8100
       SF = .5
       S1 = 1.
       Z(2) = SF * (Z(1) - Z(N+1))
       Z(1) = SF * (Z(1) + Z(N+1))
       X1 = -X1
       GO TO 8110
 8100 CONTINUE
       Z(N+1) = SF * (Z(1) - Z(2))
       Z(N+2) = 0.0
       Z(1) = SF * (Z(1) + Z(2))
       Z(2) = 0.0
       IF (MOD(ND2,2) .NE. 0) GO TO 8105
         Z(ND2+1) = SF * Z(ND2+1)
         Z(ND2+2) = SF * Z(ND2+2)
 8105  CONTINUE
       SF = .5 * SF
       S1 = -1.
 8110 CONTINUE
      C1 = 0.0
      CE = COS(X1)
      SE = SIN(X1)
      K1 = N - 1
      K0 = 3
 8200 IF (K0 .GE. K1) RETURN
        X1 = CE * C1 - SE * S1
        Y1 = CE * S1 + SE * C1
        X2 = .5 * (3. - (X1 * X1 + Y1 * Y1))
        C1 = X1 * X2
        S1 = Y1 * X2
        X1 = Z(K0) - Z(K1)
        Y1 = Z(K0+1) + Z(K1+1)
        X2 = C1 * X1 - S1 * Y1
        Y2 = C1 * Y1 + S1 * X1
        X1 = Z(K0) + Z(K1)
        Y1 = Z(K0+1) - Z(K1+1)
        Z(K0) = SF * (X1 + X2)
        Z(K0+1) = SF * (Y1 + Y2)
        Z(K1) = SF * (X1 - X2)
        Z(K1+1) = SF * (Y2 -Y1)
        K1 = K1 - 2
        K0 = K0 + 2
        GO TO 8200
      END
```

## BIBLIOGRAPHY


1. Kovacs, F.A. and T.A. Cross, Transmembrane four-helix bundle of influenza A M2 protein channel: Structural implications from helix tilt and orientation. Biophys. J., 1997. **73**: p. 2511-2517.

2. Kim, Y., et al., Solid-state NMR studies of the membrane-bound closed state of the colicin E1 channel domain in lipid bilayers. Protein Science, 1998. **7**(2): p. 342-8.

3. Opella, S.J., et al., Three-dimensional structure of the membrane-embedded M2 channel-lining segment from nicotinic acetylcholine receptors and NMDA receptors by NMR spectroscopy. Nat. Struct. Biol., 1999. **6**: p. 374-379.

4. Opella, S.J., et al., Structures of the M2 channel-lining segments from nicotinic acetylcholine and NMDA receptors by NMR spectroscopy. Nature Structural Biology, 1999. **6**(4): p. 374-379.

5. Woolf, T.B. and B. Roux, The binding site of sodium in the gramicidin A channel: Comparison of molecular dynamics with solid-state NMR data. Biophysical Journal, 1997. **72**: p. 1930-1945.

6. Cross, T.A., et al., Correlations of structure, dynamics and function in the gramicidin channel by solid-state NMR spectroscopy. Novartis Foundation Symposium, 1999. **225**: p. 4-16; discussion 16-22.

7. Kikuchi, J. and T. Asakura, Use of 13C conformation-dependent chemical shifts to elucidate the local structure of a large protein with homologous domains in solution and solid state. Journal of Biochemical and Biophysical Methods, 1999. **38**(203-208).

8. Ottiger, M. and A. Bax, Characterization of magnetically oriented phospholipid micelles for measurement of dipolar couplings in macromolecules. J. Biomol. NMR, 1998. **12**: p. 361-372.

9. Sanders, C.R., et al., Magnetically-oriented phospholipid micelles as a tool for the study of membrane-associated molecules. Prog. NMR Spectrosc., 1993. **26**: p. 431-444.

10. Glaubitz, C., G. Grobner, and A. Watts, Structural and orientational information of the membrane embedded M13 coat protein by C-13-MAS NMR spectroscopy. Biochimica Et Biophysica Acta-Biomembranes, 2000. **1463**(1): p. 151-161.

11. Almeida, F.C. and S.J. Opella, fd coat protein structure in membrane environments: structural dynamics of the loop between the hydrophobic trans-membrane helix and the amphipathic in-plane helix. Journal of Molecular Biology, 1997. **270**(3): p. 481-95.

12. Bechinger, B., M. Zasloff, and S.J. Opella, Structure and interactions of magainin antibiotic peptides in lipid bilayers: a solid-state nuclear magnetic resonance investigation. Biophysical Journal, 1992. **62**(1): p. 12-4.

13. Bechinger, B., M. Zasloff, and S.J. Opella, Structure and orientation of the antibiotic peptide magainin in membranes by solid-state nuclear magnetic resonance spectroscopy. Protein Science, 1993. **2**(12): p. 2077-84.

14. Ketchem, R.R., W. Hu, and T. Cross, High-resolution conformation of gramicidin A in a lipid bilayer by solid-state NMR. Science, 1993. **261**: p. 1457-1460.

15. Smith, R., et al., Melittin-induced changes in lipid multilayers. A solid-state NMR study. Biophysical Journal, 1992. **63**(2): p. 469-74.

16. Gregory, D.M., et al., Dipolar Recoupling NMR of biomolecular self-assemblies; inter and intrastrand distances in fibrilized Alzheimer's beta-amyloid peptide. Solid State Nuclear Magnetic Resonance, 1998. **13**: p. 149-166.

17. van_Beek, J.D., et al., Solid-State NMR determination of the secondary structure of Samia cynthia ricini silk. Nature, 2000. **406**: p. 1077-1079.

18. Batchelder, L.S., et al., Characterization of leucine side-chain reorientation in collagen fibrils by solid-state 2H NMR. Proc. Nat. Acad. Sci. USA, 1982. **79**: p. 386-389.

19. Fu, R. and T.A. Cross, Solid-state nuclear magnetic resonance investigation of protein and polypeptide structure. Annu Rev. Biophys. Biomol. Struct, 1999. **28**: p. 235-68.

20. Ramamoorthy, A., C.H. Wu, and S.J. Opella, Magnitudes and Orientations of the Principal Elements of the 1H chemical shift, 1H-15N dipolar coupling, and 15N chemical shift interaction tensors in the 15N-epsilon1-tryptophan and 15N-pi-histidine side chains determined by three-dimensional solid-state NMR spectroscopy of polycrystalline samples. J. Am. Chem. Soc., 1997. **119**: p. 10479-10486.

21. Marassi, F.M., et al., Three-Dimensional Solid-State NMR Spectroscopy is Essential for Resolution of Resonances from In-Plane Residues in Uniformly 15N-Labeled Helical Membrane Proteins in Oriented Lipid Bilayers. J. Mag. Res, 2000. **144**: p. 156-161.

22. Marassi, F.M., A. Ramamoorthy, and S.J. Opella, Complete resolution of the solid-state NMR spectrum of a uniformly 15N labeled membrane protein in phospholipid bilayers. Proc. Natl. Acad. Sci. USA, 1997. **94**: p. 8551-8556.

23. Kosen, P.A., Spin labeling of proteins. Methods in Enzymology, 1989. **177**: p. 86-121.

24. Fujiwara, T., et al., C-13-C-13 and C-13-N-15 dipolar correlation NMR of uniformly labeled organic-solids for the complete assignment of their C-13 and N-15 signals-An application to adenosine. J. Am. Chem. Soc., 1995. **117**: p. 11351-11352.

25. Andrew, E.R. and E. Szczesniak, A historical account of NMR in the solid state. Progress in Nuclear Magnetic Spectroscopy, 1995. **28**: p. 11-36.

26. Zell, M.T., et al., Two-Dimensional High-Speed CP/MAS NMR spectroscopy of polymorphs. 1. Uniformly 13C-Labeled Aspartame. J. Am. Chem. Soc, 1999. **121**: p. 1372-1378.

27. Tycko, R., Prospects for resonance assignments in multidimensional solid-state NMR spectra of uniformly labeled proteins. J. Biomol. NMR, 1996. **8**: p. 239-251.

28. Aggarwal, B.B., Apoptosis and nuclear factor-kappaB: A tale of association and disassociation. Biochemical Pharmacology, 2000. **60**(8): p. 1033-1039.

29. Hinuma, S., H. Onda, and M. Fuijino, The quest for novel bioactive peptides utilizing orphan seven-transmembrane-domain receptors. J. Mol. Med., 1999. **77**(6): p. 495-504.

30. Rayan, A., et al., A novel computation method for predicting the transmembrane structure of G-protein coupled receptors: application to human C5aR and C3aR. Receptors Channels, 2000. **7**(2): p. 121-137.

31. Sun, J., et al., Identification of ligand effector binding sites in transmembrane regions of the human G protein-coupled C3a receptor. Protein Sci, 1999. **8**(11): p. 2304-2311.

32. Feng, Y.H. and S.S. Karnik, Role of transmembrane helix IV in G-protein specificity of the angiotensin II type 1 receptor. J. Biol. Chem, 1999. **274**(50): p. 35546-35552.

33. Palczewski, K., et al., Crystal Structure of Rhodopsin: A G Protein-Coupled Receptor. Science, 2000. **289**(5480): p. 739-745.

34. Gershengorn, M.C. and R. Osman, Minireview: Insights into G Protein-Coupled Receptor Function using Molecular Models. Endocrinology, 2001. **142**(1): p. 2-10.

35. Leach, A.R., R.A. Bryce, and A.J. Robinson, Synergy between combinatorial chemistry and de novo design. J. Mol. Graph. Model, 2000. **18**(4-5): p. 358-367, 526.

36. Eden, M. and M.H. Levitt, Excitation of carbon-13 triple quantum coherence in magic-angle-spinning NMR. Chem. Phys. Lett., 1998. **293**: p. 173-179.

37. Klug, C.A. and J. Schaefer, Extension of CEDRA to homonuclear coherence transfer. J. Magn. Reson. A, 1996. **122**: p. 251-253.

38. Tomaselli, M., et al., Nuclear magnetic resonance polarization and coherence echoes in static and rotating solids. J. Chem. Phys., 1996. **105**: p. 10672-10681.

39. Luy, B. and S. Glaser, Analytical Polarization and Coherence Transfer Functions for Three Dipolar Coupled Spins 1/2 under Cylindrical Mixing Conditions. J. Mag. Res., 2000. **142**: p. 280-287.

40. Taylor, D.M. and A. Ramamoorthy, Analysis of Dipolar-Coupling-Mediated Coherence Transfer in a Homonuclear Two Spin-1/2 Solid-State System. J. Mag. Res., 1999. **141**: p. 18-28.

41. Ernst, R.R., G. Bodenhausen, and A. Wokaun, Principles of Nuclear Magnetic Resonance in One and Two Dimensions. 1987, Oxford: Clarendon Press.

42. Ross, A., et al., Automation of NMR measurements and data evaluation for systematically screening interactions of small molecules with target proteins. J. Biomol. NMR, 2000. **16**(2): p. 139-46.

43. Brunner, E., et al., Molecular alignment of proteins in bicellar solutions: quantitative evaluation of effects induced in 2D COSY spectra. Biochem. Biophys. Res. Comm., 2000. **272**(3): p. 694-698.

44. Struppe, J., J.A. Whiles, and R.R. Vold, Acidic phospholipid bicelles: a versatile model membrane system. Biophys. J., 2000. **78**(1): p. 281-189.

45. Sanders, C.R. and K. Oxenoid, Customizing model membranes and samples for NMR spectroscopic studies of complex membrane proteins. Biochimica et Biophysica Acta, 2000. **1508**: p. 129-145.

46. Warchawski, D.E., et al., Solid-state NMR for the study of membrane systems: the use of anisotropic interactions. Biochimie, 1998. **80**: p. 437-450.

47. Losonczi, J.A., F. Tian, and J.H. Prestegard, Nuclear Magnetic Resonance studies of the N-terminal fragment of adenosine diphosphate ribosylation factor 1 in micelles and bicelles: influence of N-myristoylation. Biochemistry, 2000. **39**(13): p. 3804-3816.

48. Chandrakumar, N. and S. Subramanian, Some Aspects of Coherence Transfer by Isotropic Mixing. J. Magn. Reson., 1985. **62**: p. 346-349.

49. Ramamoorthy, A. and N. Chandrakumar, Comparison of the Coherence-Transfer Efficiencies of Laboratory- and Rotating-Frame Experiments. J. Magn. Reson., 1992. **100**: p. 60-68.

50. Miao, X., X. Han, and J. Hu, Application of Product Operator Formalism to the Strongly Coupled Spin (I=1/2) Systems. Science in China (Series A), 1993. **36**(10): p. 1199-1211.

51. Miao, X. and C. Ye, Application of the product operator formalism to spin (I=1/2) systems under a radio-frequency irradiation. Mol. Phys., 1997. **90**(4): p. 499-514.

52. Nakai, T. and C.A. McDowell, Product operator theory for ABX spin systems and its application to H-C-C INEPT NMR experiments. Mol. Phys., 1994. **81**: p. 337-358.

53. Sorenson, O.W., et al., Product operator formalism for the description of NMR pulse experiments. Progress in NMR Spectroscopy, 1983. **16**: p. 163-192.

54. Vega, S., Fictitious spin 1/2 operator formalism for multiple quantum NMR. J. Chem. Phys., 1978. **68**(12): p. 5518-5527.

55. Vega, S. and A. Pines, Operator formalism for double quantum NMR. J. Chem. Phys., 1977. **66**(12): p. 5624-5644.

56. Tycko, R. and G. Dabbagh, Measurement of nuclear magnetic dipole- dipole couplings in magic angle spinning NMR. Chem. Phys. Lett, 1990. **173**: p. 461-465.

57. Gregory, D.M., et al., Windowless Dipolar Recoupling - the Detection of Weak Dipolar Couplings between Spin-1/2 Nuclei with Large Chemical-Shift Anisotropies. Chem. Phys. Lett., 1995. **246**(6): p. 654-663.

58. Fujiwara, T., et al., Dipolar HOHAHA under MAS conditions for solid-state NMR. Chem. Phys. Lett., 1993. **212**: p. 81-84.

59. Bennett, A.E., J.H. Ok, and R.G. Griffin, Chemical shift correlation spectroscopy in rotating solids: Radio Frequency-driven Dipolar Recoupling and longitudinal exchange. J. Chem. Phys., 1992. **96**: p. 8624-8627.

60. Mehring, M., Principles of High Resolution NMR in Solids. 1983, New York: Springer-Verlag.

61. Gerstein, B.C. and C.R. Dybowski, Transient Techniques in Nmr of Solids : An Introducton to Theory and Practice. 1985, Orlando: Academic Press.

62. Slichter, C.P., Principles of Magnetic Resonance. 3 ed. Springer Series in Solid-State Sciences, ed. M. Cardona, et al. 1996, Berlin: Springer-Verlag.

63. Stejskal, E.O. and J.D. Memory, High Resolution NMR in the Solid State: Fundamentals of CP/MAS. 1994, New York, NY: Oxford University Press.

64. Cerf, C., NMR Spectroscopy: From Quantum Mechanics to Protein Spectra. Concepts in Magn. Reson., 1997. **9**(1): p. 17-41.

65. Smith, S.A., W.E. Palke, and J.T. Gerig, The Hamiltonians of NMR: Part I. Concepts in Magn. Reson., 1992. **4**: p. 107-144.

66. Smith, S.A., W.E. Palke, and J.T. Gerig, The Hamiltonians of NMR: Part II. Concepts in Magn. Reson., 1992. **4**: p. 181-204.

67. Smith, S.A., W.E. Palke, and J.T. Gerig, The Hamiltonians of NMR: Part III. Concepts in Magn. Reson., 1992. **5**: p. 151-177.

68. Smith, S.A., W.E. Palke, and J.T. Gerig, The Hamiltonians of NMR: Part IV. Concepts in Magn. Reson., 1992. **6**: p. 137-162.

69. Smith, S.O., K. Aschheim, and M. Groesbeek, Magic angle spinning NMR spectroscopy of membrane proteins. Q. Rev. Biophys., 1996. **29**: p. 395-559.

70. Sakurai, J.J., Modern Quantum Mechanics, ed. S.F. Tuan. 1995, Reading, Massachusetts: Addison-Wesley.

71. Brink, D.M. and G.R. Satchler, Angular Momentum. Third ed. 1994, Oxford, England: Clarendon Press.

72. Rose, M.E., Elementary Theory of Angular Momentum. 1995, New York, NY: Dover.

73. Wuthrich, K., NMR of Proteins and Nucleic Acids. 1986, New York: Wiley Interscience.

74. Pines, A., M.G. Gibby, and J.S. Waugh, Proton-enhanced NMR of dilute spins in solids. J. Chem. Phys, 1973. **59**: p. 569-590.

75. Meier, B.H. and W.L. Earl, Excitation of multiple quantum transitions under magic angle spinning conditions: Adamantane. J. Chem. Phys, 1986. **85**: p. 4905-4911.

76. Meier, B.H. and W.L. Earl, A double-quantum filter for rotating solids. J. Am. Chem. Soc, 1987. **109**: p. 7937-7942.

77. Colombo, M.G., B.H. Meier, and R.R. Ernst, Rotor-driven spin diffusion in natural-abundance 13 C spin systems. Chem. Phys. Lett., 1988. **146**: p. 189-196.

78. Raleigh, D.P., M.H. Levitt, and R.G. Griffin, Rotational Resonance in Solid State NMR. Chem Phys Lett, 1988. **146**(71-76).

79. Levitt, M.H., T.G. Oas, and R.G. Griffin, Rotary resonance recoupling in heteronuclear spin pair systems. Isr. J. Chem, 1988. **28**: p. 271-282.

80. Robyr, P., B.H. Meier, and R.R. Ernst, Radio-frequency-driven nuclear spin diffusion in solids. Chem. Phys. Lett, 1989. **162**: p. 417-423.

81. Levitt, M.H., et al., Theory and simulation of homonuclear spin pair systems in rotating solids. J. Chem. Phys, 1990. **92**: p. 6347-6364.

82. Gan, Z.H. and D.M. Grant, Rotational resonance in a spin-lock field for solid state NMR. Chem. Phys. Lett., 1990. **168**(304-308).

83. Bennett, A.E., et al., Homonuclear correlation spectroscopy in rotating solids. Chem. Phys. Lett, 1992. **96**: p. 8624-8627.

84. Ok, J.H., et al., Homonuclear correlation spectroscopy in rotating solids. Chem. Phys. Lett., 1992. **197**(389-396).

85. Nielsen, N.C., et al., Enhanced double-quantum nuclear magnetic resonance in spinning solids at rotational resonance. J. Chem. Phys, 1992. **96**: p. 5668-5677.

86. Gullion, T. and S. Vega, A simple magic angle spinning NMR experiment for the dephasing of rotational echoes of dipolar coupled homonuclear spin pairs. Chem. Phys. Lett., 1992. **194**: p. 423-428.

87. Tycko, R. and S.O. Smith, Symmetry principles in the design of pulse sequences for structural measurements in magic angle spinning nuclear magnetic resonance. J. Chem. Phys., 1993. **98**: p. 932-943.

88. Sodickson, D.K., et al., Broad band dipolar recoupling in the nuclear magnetic resonance of rotating solids. J. Chem. Phys., 1993. **98**: p. 6742-6748.

89. Nielsen, N.C., F. Creuzet, and R.G. Griffin, Rotationally enhanced exchange of Zeeman order via two-dimensional switched-speed spinning NMR. J. Magn. Reson. A, 1993. **103**: p. 245-252.

90. Spencer, R.G.S., et al., Rotational resonance with multiple-pulse scaling in solid-state nuclear- magnetic-resonance. J. Chem. Phys, 1994. **100**: p. 5533-5545.

91. Nakai, T. and C.A. McDowell, Residual chemical-shift effects in spin-echo NMR powder patterns of homonuclear dipolar-coupled spins. Chem. Phys. Lett., 1994. **217**: p. 234-238.

92. Nakai, T. and C.A. McDowell, Characterization of homonuclear spin pairs from two-dimensional spin-echo NMR powder patterns. J. Am. Chem. Soc., 1994. **116**: p. 6373-6383.

93. Bennett, A.E., R.G. Griffin, and S. Vega, Recoupling of homo- and heteronuclear dipolar interactions in rotating solids. NMR Basic Principles Prog., 1994. **12**: p. 1-77.

94. Tomaselli, M., et al., An rf-driven nuclear spin-diffusion experiment using zero-angle sample spinning. Chem. Phys. Lett., 1994. **225**: p. 131-139.

95. Karlsson, T., et al., Rotational resonance echoes in the nuclear magnetic resonance of spinning solids. Chem. Phys. Lett., 1995. **247**: p. 534-540.

96. Sun, B.Q., et al., Internuclear distance measurements in solid state nuclear magnetic resonance: Dipolar recoupling via rotor synchronized spin locking. J. Chem. Phys., 1995. **102**: p. 702-707.

97. Robyr, P., et al., RF-driven and proton-driven NMR polarization transfer for investigating local order. An application to solid polymers. Mol. Phys., 1995. **84**: p. 995-1020.

98. Eden, M., Y.K. Lee, and M.H. Levitt, Efficient simulation of periodic problems in NMR. Application to decoupling and rotational resonance. J. Magn. Reson. A, 1996. **120**: p. 56-71.

99. Sachleben, J.R., V. Frydman, and L. Frydman, Dipolar determinations in solids by relaxation-assisted NMR recoupling. J. Am. Chem. Soc., 1996. **118**: p. 9786-9787.

100. Karlsson, T. and M.H. Levitt, Longitudinal rotational resonance echoes in solid state nuclear magnetic resonance: Investigation of zero quantum spin dynamics. J. Chem. Phys., 1998. **109**: p. 5493-5507.

101. Hohwy, M., et al., Broadband dipolar recoupling in the nuclear magnetic resonance of rotating solids: A compensated C7 pulse sequence. J. Chem. Phys., 1998. **108**: p. 2686-2694.

102. Kiihne, S., et al., Distance measurements by dipolar recoupling two-dimensional solid-state NMR. J. Phys. Chem. A, 1998. **102**(2274-2282).

103. Menger, E.M., S. Vega, and R.G. Griffin, Observation of carbon- carbon connectivities in rotating solids. J. Am. Chem. Soc, 1986. **108**: p. 2215-2218.

104. Griffiths, M., et al., Dipolar correlation NMR spectroscopy of a membrane protein. J. Am. Chem. Soc., 1994. **116**: p. 10178-10181.

105. Sandstrom, D. and M.H. Levitt, Structure and molecular ordering of a nematic liquid crystal studied by natural-abundance double-quantum 13C NMR. J. Am. Chem. Soc., 1996. **118**(6966-6974).

106. Weliky, D.P. and R. Tycko, Determination of peptide conformations by two-dimensional magic angle spinning NMR exchange spectroscopy with rotor synchronization. J. Am. Chem. Soc., 1996. **118**: p. 8487-8488.

107. Tycko, R., D.P. Weliky, and A.E. Berger, Investigation of molecular structure in solids by two-dimensional NMR exchange spectroscopy with magic angle spinning. J. Chem. Phys., 1996. **105**: p. 7915-7930.

108. Costa, P.R., et al., Solid-state NMR measurement of Psi in peptides: A NCCN 2Q-heteronuclear local field experiment. Chem. Phys. Lett., 1997. **280**: p. 95-103.

109. Hong, M., et al., Coupling amplification in 2D MAS NMR and its application to torsion angle determination in peptides. J. Magn. Reson., 1997. **129**: p. 85-92.

110. Sun, B.Q., et al., 3D 15N-13C-13C chemical shift correlation spectroscopy in rotating solids. J. Am. Chem. Soc., 1997. **119**: p. 8540-8546.

111. Feng, X., et al., Direct determination of a molecular torsional angle in the membrane protein rhodopsin by solid-state NMR. J. Am. Chem. Soc, 1997. **119**: p. 6853-6857.

112. Middleton, D.A., et al., The conformation of an inhibitor bound to the gastric proton pump. FEBS Lett., 1997. **410**: p. 269-274.

113. Bennett, A.E., D.P. Weliky, and a.R. Tycko, Quantitative conformational measurements in solid state NMR by constant-time homo-nuclear dipolar recoupling. J. Am. Chem. Soc., 1998. **120**: p. 4897-4898.

114. Fujiwara, T., et al., Multidimensional solid-state nuclear magnetic resonance for determining the dihedral angle from the correlation of C-13-H-1 and C-13-C-13 dipolar interactions under magic-angle spinning conditions. J. Chem. Phys., 1998. **109**: p. 2380-2393.

115. Burum, D.P. and R.R. Ernst, Net polarization transfer via a J-ordered state for signal enhancement of low-sensitivity nuclei. J. Magn. Reson., 1980. **39**(1): p. 163-168.

116. Cross, T.A. and S.J. Opella, Solid-state NMR structural studies of peptides and proteins in membranes. Curr. Opin. Struct. BIol., 1994. **4**: p. 574-581.

117. Ramamoorthy, A., F.M. Marassi, and S.J. Opella, Applications of multi-dimensional solid-state NMR spectrosocpy to membrane proteins, in Proceedings of the International School of Biological Magnetic Resonance, Second Course, Dynamics and the Problem of Recognition in Biological Macromolecules, O. Jardetsky and J. Lefeure, Editors. 1996, Plenum Press: New York. p. 237-255.

118. Sanders, C.R. and R.S. Prosser, A model membrane system for all seasons. Structure, 1998. **6**: p. 1227-1234.

119. Ottiger, M. and A. Bax, Determination of relative N-H-N N-C9, C-alpha-C9, and C(alpha)-H-alpha effective bond lengths in a protein by NMR in a dilute liquid crystalline phase. J. Am. Chem. Soc., 1998. **120**: p. 12334-12341.

120. Long, H.W. and R. Tycko, Biopolymer conformational distributions from solid-state NMR: Alpha-helix and 3(10)-helix contents of a helical peptide. J. Am. Chem. Soc., 1998. **120**: p. 7039-7048.

121. Brueschweiler, R., Dipolar Averaging in NMR Spectroscopy: From polarization transfer to cross relaxation. Prog. NMR Spectrosc., 1998. **32**: p. 1-19.

122. Born, M. and E. Wolf, Principles of Optics. 7th expanded ed. 1999, London: Cambridge University Press.

123. Williams, T. and C. Kelley, Gnuplot; a plotting program, . 2000, GNU Consortium.

124. Wolfram, S., Mathematica, . 2000, Wolfram Research.

125. Luy, B. and S. Glaser, Superposition of Scalar and Residual Dipolar Couplings: Analytical Transfer Functions for Three Spins 1/2. Journal of Magnetic Resonance, 2000. **148**(1): p. 169-181.

126. Bennett, A.E., et al., Heteronuclear Decoupling in Rotating Solids. Journal of Chemical Physics, 1995. **103**(16): p. 6951-6958.

127. Lee, D.K., R.J. Wittebort, and A. Ramamoorthy, Characterization of N-15 chemical shift and H-1-N-15 dipolar coupling interactions in a peptide bond of uniaxially oriented and polycrystalline samples by one-dimensional dipolar chemical shift solid-state NMR spectroscopy. Journal of the American Chemical Society, 1998. **120**(34): p. 8868-8874.

128. Lee, D.K., J.S. Santos, and A. Ramamoorthy, Application of one-dimensional dipolar shift solid-state NMR spectroscopy to study the backbone conformation of membrane- associated peptides in phospholipid bilayers. Journal of Physical Chemistry B, 1999. **103**(39): p. 8383-8390.

129. Lee, D.K., J.S. Santos, and A. Ramamoorthy, Nitrogen-15 chemical shift anisotropy and H-1-N-15 dipolar coupling tensors associated with the phenylalanine residue in the solid state. Chemical Physics Letters, 1999. **309**(3-4): p. 209-214.

130. Schmidt-Rohr, K. and H.W. Spiess, Multidimensional Solid-State NMR and Polymers. 1994, London: Academic Press.