# GAMMA

# COSY Examples

$$\hat{\hat{\Gamma}}$$

| | |
|---|---|
| Author: | Scott A. Smith |
| Date: | August 19, 1999 |

# Table of Contents

# 1 COSY

## 1.1 Introduction

The pulse sequence below is the simplest of the COSY experiments.

***Basic COSY Pulse Sequence***

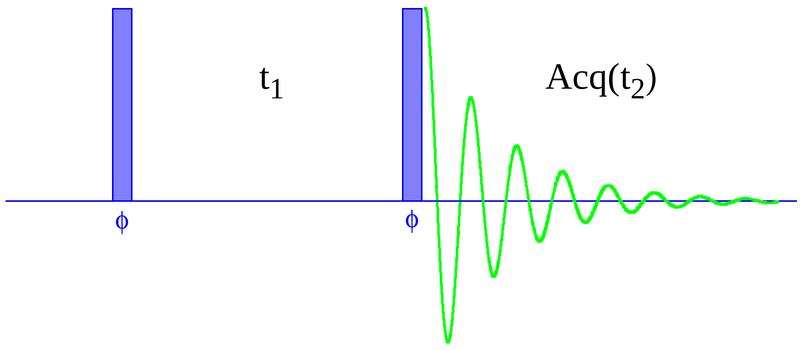


***Figure 0-1*** **A simple COSY pulse sequence .**

The first pulse generates magnetization in the xy-plane. Subsequently, the magnetization is frequency labeled during $t_1$ and the final pulse produces the observable signal. The chemical shifts of the spin system show up as frequencies during the acquisition and these are modulated by the frequency labeling during $t_1$ to all spins with which they are spin coupled.

The following set of programs which simulate COSY-experiments are discussed in this document:

**Table 1: COSY Example Programs**

| Example | Page | Pulse[a] | Relaxation | System | Workup[b] |
|---|---|---|---|---|---|
| 1. COSYNMRi | 36 | Ideal | No | Homonuclear | NMR2 |
| 2. COSYFelix | 46 | Ideal | No | Homonuclear | Felix |
| 3. COSYTPPI | 54 | Ideal | No | Homonuclear | Felix/TPPI |
| 4. COSYRuSH | 61 | Real | No | Heteronuclear | Felix |

a. “Ideal” pulses are “infinitely” short so no relaxation effects can be considered during such sequence steps.
b. Workups labeled “Open” allow the user to choose the output type while the program is running. The output types currently available in GAMMA are FrameMaker, Felix, NMRi, and MATLAB

Please note that these examples are meant as simple tutorials in getting COSY programs built. They are typically NOT the most elegant nor the most versatile. However they should provide a wide basis on which robust simulation programs can be built. To obtain these and more COSY programs see the GAMMA WWW site at http::/gamma.magnet.fsu.edu.

## 1.2 COSY- NMRi, Ideal Pulses

### 1.2.1 Description

This example performs a simple COSY simulation. The pulse sequence diagram is repeated with density matrix labels used to coincide with variable names in the program code.

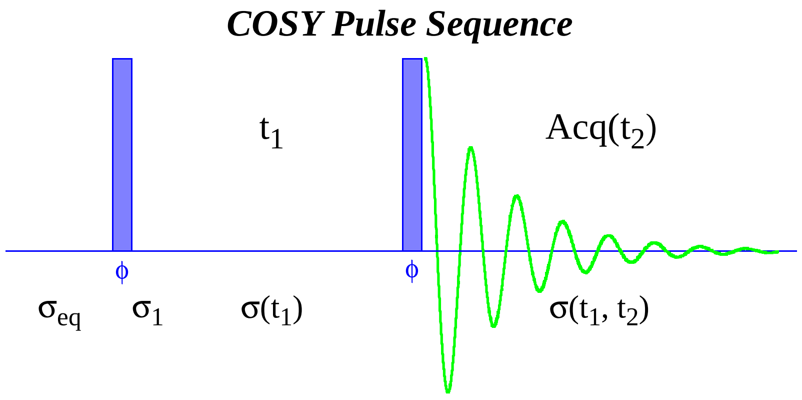


***Figure 0-2*** **A simple COSY pulse sequence. Density operators label key points in simulation of this exepriment.**

The code itself will not the most efficient in producing a COSY simulation, efficiency was sacrificed (slightly) for code clarity. No relaxation effects will be considered and the pulses taken as ideal. The resulting output file is produced in NMRi format and subsequently processed with that program. The size of the working data set is set for 1K x 1K and the spin system only has three protons. Although the run time is in minutes, although the data workup is quite lengthy.

### 1.2.2 Program

```
/* cosy1.cc ******************************************************************-*-c++- *-
**                                                                                    **
**                      Example program for the GAMMA Library                         **
**                                                                                    **
** This program simulates a basic COSY experiment.                                    **
** No phase cycling is performed and all input are                                    **
** kept positive to avoid the need for quadrature                                     **
** detection in t1.                                                                   **
**                                                                                    **
************************************************************************************** */

#include <gamma.h>
main()
//                    DEFINE SYSTEM & NMR PARAMETERS

  const double dt1 = 0.001;                          // t1 time increment
  const double dt2 = 0.001;                          // t2 time increment
  const int t1pts = 1024;                            // points on t1 axis
  const int t2pts = 1024;                            // points on t2 axis
  spin_system sys;                                   // define the system, read in
  sys.read("cosy.1.sys");                            // from disk
//                    SET UP NECESSARY VARIABLES

  block_1D tmp(t2pts);                               // 1D-data block storage
  block_2D data(t1pts,t2pts);                        // 2D-data matrix storage
  gen_op H = Hcs(sys)+ HJw(sys);                     // Hamiltonian, weak coupling
  gen_op detect = Fm(sys);                           // F- for detection operator
  gen_op sigma0, sigma1, sigma;                      // working density matrices
//                    APPLY PULSE SEQUENCE

  sigma0 = sigma_eq(sys);                            // equilibrium density matrix
  sigma1 = Iypuls(sys, sigma0, 90);                  // apply first 90 y-pulse
  for (int t1=0; t1<t1pts; t1++)                     // loop over t1 points
   {
   sigma=evolve(sigma1, H, t1*dt1);                  // evolution during t1
   sigma=Iypuls(sys, sigma, 90);                     // apply second 90 y-pulse
   FID(sigma,detect,H,dt2,t2pts,tmp);                // acquisition
   data.put_block(t1, 0, tmp);                       // store it in a 2D block
   }
NMRi("cosy.1.dat", data);                            // output for NMRi workup
}
```

### 1.2.3 Discussion

**Define System and NMR Parameters:** The first four lines specify the dwell times and number of points to use on the $t_1$ and $t_2$ axes. The spin system used was a three spin homonuclear system with weak coupling, read in from disk file cosy1.sys (see next section).

**Set up the Necessary Variables:** For this simulation two data blocks are allocated. The first is a 1D-block called tmp for temporary FID storage. The second is a 2D-block called data into which each 1D block is successively placed. The Hamiltonian is set for weak scalar coupling (under the secular approximation). The detector is set to $F_- = F_x - iF_y$ so as to produce a complex data set along the t2 axis. Three density matrices are specified, coinciding with the three matrices in the previous pulse sequence figure.

**Apply the Pulse Sequence:** The program sets the initial density matrix, $\sigma_0$, to the equilibrium density matrix. A ninety degree pulse is then applied to it to produce $\sigma_1$. As can be seen from the applied pulse sequence, the density matrix $\sigma_1$ is constant for all mixing times. It's computation is thus performed prior to beginning any looping over $t_1$ times. Following this the looping begins over all $t_1$ points. The first loop step is to evolve $\sigma_1$ for the $t_1$ time to produce $\sigma(t_1)$. The second 90 pulse of the sequence is then applied and the xy-magnetization determined. This is an acquisition step which fills the data block tmp up with t2pts FID points. The final step is to take that individual 1D-block and insert it into a 2D-block, data. Then data is the 2D-data set, output on the last step of the program in NMRi format.

### 1.2.4 Spin System

The spin system file, cosy.1.sys, is reproduced below.

```
SysName  (2) : COSY3  - Name of the Spin System
NSpins   (0) : 3      - Number of Spins in the System
Iso(0)   (2) : 1H     - Spin Isotope Type
Iso(1)   (2) : 1H     - Spin Isotope Type
Iso(2)   (2) : 1H     - Spin Isotope Type
v(0)     (1) : 200.0  - Chemical Shifts in Hz
v(1)     (1) : 100.0  - Chemical Shifts in Hz
v(2)     (1) : 40.0   - Chemical Shifts in Hz
J(0,1)   (1) : 10.0   - Coupling Constants in Hz
J(1,2)   (1) : 15.0   - Coupling Constants in Hz
Omega     (1) : 400   - Spectrometer Frequency in MHz (1H based)
```

### 1.2.5 Results

The 2D-data file produced was processed with NMRi, specifically NMR2[1,2]. The data set was manipulated within NMR2 by a link (macro resulting from the linking) of the following commands:

| | |
|---|---|
| SP | apodize in t2 with Sine bell Phased function (0.5, 1, 6) |
| FT | Fourier Transformation in t2 |
| TP | Transpose the entire data set |
| EI | set Imaginary part to 0 since it is redundant with real data |
| SP | apodize in t1 with Sine bell Phased function (0.5, 1, 6) |
| FT | Fourier transformation in t1 |
| TP | Transpose the entire data set back so F2 is horizontal |

Note that the comments to the right are only for description of the two letter commands used to create the link. The link is produced in the following manner.

```
1. CM  - go into Command Mode
       1a.) M- Modify/create link.
            1a.-1)                    - Supply a link name.
            1a.-2)                    - Type two letter link commands (above).
            1a.-3)                    - Hit return with no command to end the link.
       1b.) X- Execute (in a dry run first) the link
            1b-1)                     - SP parameters are 0.5, 1, 6
            1b-2)                     - EI parameter is 0
            1b-3)                     - SP parameters are 0.5, 1, 6 (same as t2 axis)
2. DM  - go into Display Mode. (Note all DM commands not shown in window)
       1a.) P- Define contour levels.
```

1. NMR1 and NMR2 (jointly called NMRi) are products of New Methods Research, Inc. in Syracuse, New York. Phone: (315) 424-0329. These programs are designed specifically for the workup of NMR data.
2. This example produces a 1K by 1K complex output file. Data processing will be time consuming and limited by computer system memory and/or disk space availability. Switching from 1024 to 256 in the program, both time and memory requirements for the simulation and data processing will be greatly reduced.

The COSY spectrum from simulation cosy.1. This is the NMR2 output format

```
                Parameters PLP.H01 - NMR2 3.95
               Data File: /data4/sosi/shit.trf

      1st Dim Full SW: 1000 Hz                  Size: 1024
      2nd Dim Full SW: 1000 Hz                  Spectra: 1024
      F2 Obs Freq: 400.0 MHz                    Data Type: quadrature
      F1 Obs Freq: 400.0 Mhz                    Data are not transposed.
      Lock Freq: 0.0 Mhz                        Mode: Phase Sens. TPPI.
      Acquisitions: 0                           FDATA(512) : 0.0

      Apodization Function SP.                  Apodization Function SP.
      ->F2 Q1 = 0.5                            ->F1 Q1 = 0.5
      ->F2 Q2 = 1.0                            ->F1 Q2 = 1.0
      ->F2 Q3 = 6.0                            ->F1 Q3 = 6.0

     F2 Zero Fills: 0                           F1 Zero Fills: 0
     F2 Phase: (0.00,0.00)                      F1 Phase: (0.00,0.00)
```

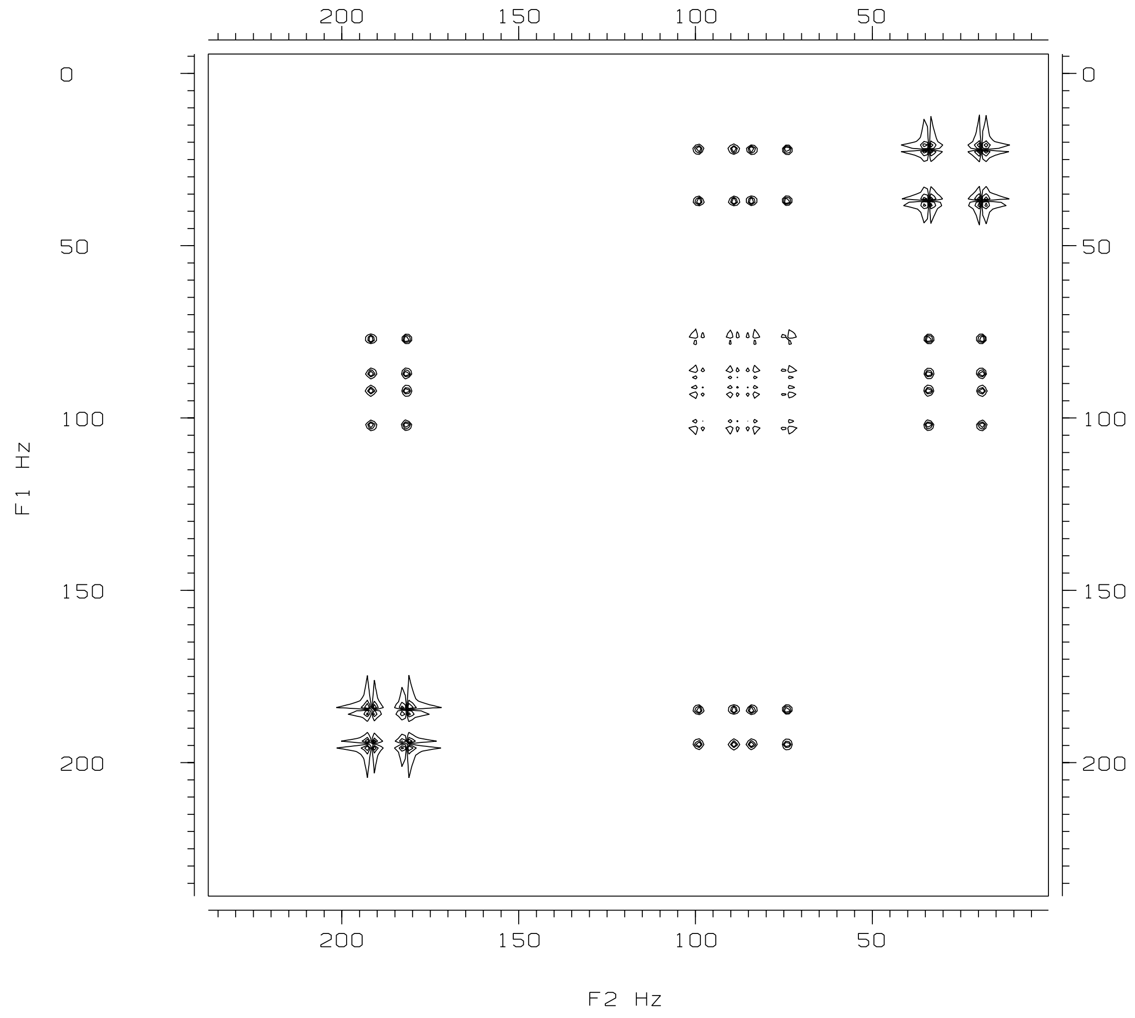


*Figure 0-3* **Simulated COSY spectrum output from NMR2, the result of COSY1.cc. The plot was placed into eps format via HPGL output from NMR2 & the filter hpgltoeps provided by FrameMaker (used for this doucment)**

## 1.3 COSY- Felix, Ideal Pulses

### 1.3.1 Description

This example again performs a basic COSY simulation. The pulse sequence diagram is repeated with density matrix labels used to coincide with variable names in the program code.

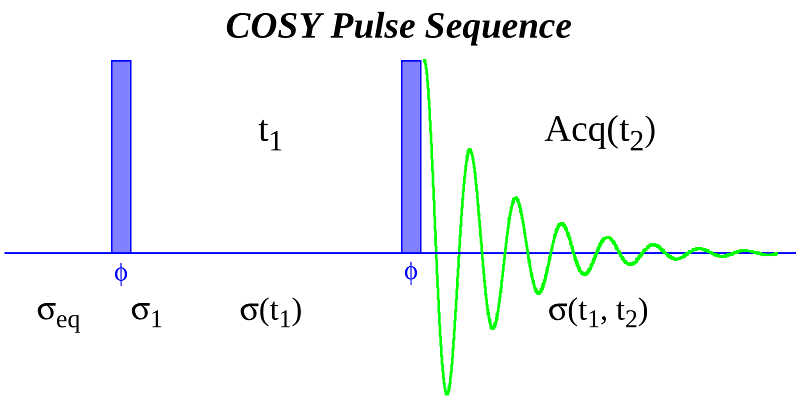


***Figure 0-4*** **A simple COSY pulse sequence. Density operators label key points in simulation of this exepriment.**

The code itself is written to be a little more efficient than in the previous example. No relaxation effects will be considered and the pulses taken as ideal. The resulting output file is produced in Felix[1] format and subsequently processed with that program. There are two differences of note between this example and the previous one. First the data processing here provides a clear example of how to work up simulated data with the Felix program (rather than NMR2 used previously). Secondly, use is made of a pulse propagator rather than recomputing the pulses for each step and each loop of the pulse sequence. Mathematically this is given by

$$\sigma_1 = U\sigma_{eq}U^{-1} \qquad \text{and} \qquad \sigma'(t_1) = U\sigma(t_1)U^{-1}$$

where $U$ is the 90-y pulse propagator given by

$$U = exp[-(iF_y\pi)/2] \ .$$

1. Felix is an NMR data processing program by Hare Research, Inc.

## 1.3.2 Program

```
/* cosy2.cc *******************************************************************-*-c++-*-
**                                                                                   **
**                    Example program for the GAMMA Library                          **
**                                                                                   **
** This program simulates a basic COSY experiment.                                   **
** No phase cycling is performed and all input are                                   **
** kept positive to avoid the need for quadrature                                    **
** detection in t1.                                                                  **
**                                                                                   **
**************************************************************************************/

#include <gamma.h>
main()
{
//                    DEFINE SYSTEM & NMR PARAMETERS
 const double dt1 = 0.0015;                         // t1 time increment
 const double dt2 = 0.0015;                         // t2 time increment
 const int t1pts = 512;                             // points on t1 axis
 const int t2pts = 512;                             // points on t2 axis
 spin_system sys;                                   // define the system, read in
 sys.read("cosy.2.sys");                            // from disk
//                    SET UP NECESSARY VARIABLES
 block_1D data(t2pts);                              // 1D-data block storage
 gen_op H = Hcs(sys)+ HJw(sys);                     // Hamiltonian, weak coupling
 gen_op detect = Fm(sys);                           // F- for detection operator
 gen_op y_pulse = Iypuls_U(sys, 90);                // 90 y-pulse propagator
 gen_op sigma0, sigma1, sigma;                      // working density matrices
//                    APPLY PULSE SEQUENCE
 sigma0 = sigma_eq(sys);                            // equilibrium density matrix
 sigma1 = evolve(sigma0, y_pulse);                  // apply first 90 y-pulse
 File fp;
 fp.open("felix.dat",io_writeonly,a_create);
 for (int t1=0; t1<t1pts; t1++)                     // loop over t1 points
 {
  sigma = evolve(sigma1, H, t1*dt1);                // evolution during t1
  evolve_ip(sigma, y_pulse);                        // apply second 90 y-pulse
  FID(sigma,detect,H,dt2,t2pts,data);               // acquisition
  Felix(fp, data);
}
 fp.close();
}
```

## 1.3.3 Discussion

**Define System and NMR Parameters:** The first four lines specify the dwell times and number of points to use on the $t_1$ and $t_2$ axes. The spin system used was a three spin homonuclear system with weak coupling, read in from disk file cosy.2.sys (see next section).

**Set up the Necessary Variables:** For this simulation only one data block is allocated (unlike the two in the previous example). The Hamiltonian is set for weak scalar coupling (under the secular approximation). The detector is set to $F_- = F_x - iF_y$ so as to produce a complex data set along the t2 axis. The next operator is a propagator for a y-pulse of 90 degrees. Since it is repeatedly used in the pulse sequence it is computationally efficient to produce it outside of the loop. Three density matrices are specified, coinciding with the three matrices in the previous pulse sequence figure.

**Apply the Pulse Sequence:** The program sets the initial density matrix, $\sigma_0$, to the equilibrium density matrix. A ninety degree pulse is applied using the y-pulse propagator to produce $\sigma_1$. As can be seen from the applied pulse sequence, the density matrix $\sigma_1$ is constant for all mixing times. It's computation is thus performed prior to beginning any looping over $t_1$ times. Following this the looping begins over all $t_1$ points. Just before the loop a file is opened called "felix.dat" for the spectral output. The first loop step is to evolve $\sigma_1$ for the $t_1$ time to produce $\sigma(t_1)$. The second 90 pulse of the sequence is then applied and the xy-magnetization determined. This is an acquisition step which fills the data block data up with t2pts FID points. The final step in the loop is to write this block out to the file "felix.dat" in Felix format. The last line of the program simply closes the file.

### 1.3.4 Spin System

The spin system file, cosy2.sys, is virtually identical to the one used in the previous simulation. It is reproduced below.

|  |  |  |
|---|---|---|
| **SysName** | **(2) : COSY3** | **- Name of the Spin System** |
| **NSpins** | **(0) : 3** | **- Number of Spins in the System** |
| **Iso(0)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(1)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(2)** | **(2) : 1H** | **- Spin Isotope Type** |
| **v(0)** | **(1) : 200.0** | **- Chemical Shifts in Hz** |
| **v(1)** | **(1) : 100.0** | **- Chemical Shifts in Hz** |
| **v(2)** | **(1) : 40.0** | **- Chemical Shifts in Hz** |
| **J(0,1)** | **(1) : 10.0** | **- Coupling Constants in Hz** |
| **J(0,2)** | **(1) : 0.0** | **- Coupling Constants in Hz** |
| **J(1,2)** | **(1) : 15.0** | **- Coupling Constants in Hz** |
| **Omega** | **(1) : 400** | **- Spectrometer Frequency in MHz (1H based)** |

### 1.3.5 Results

Workup of the simulated data was performed in Felix. A text editor was used to create the following Felix macro file, called cosy.2.mac (the comments off to the side are not part of the macro) -

```
cl                                    ! insure any data files are closed
cmx                                   ! insure any matrix files are closed
get ‘Enter matrix name: ‘ mname       ! ask user for matrix
mat &mname write                      ! open matrix with write ability
get ‘Enter FID name: ‘ dname          ! ask user for fid file name
lb 2                                  ! set line broadening
for row 1 512                         ! loop through all 512 rows
re &dname                             ! read FID from file
em                                    ! apodize
zf 1024                               ! zero fill
ft                                    ! fourier transform
rev                                   ! reverse spectrum so frequencies proper
si 512                                ! reduce the size back to 512
sto 0 &row                            ! store block in matrix row
ty row = &row                         ! type row is complete
next                                  ! get next row
for col 1 512                         ! begin column processing
loa &col 0                            ! retrieve column from matrix
em                                    ! apodize
zi                                    ! zero imaginaries (real data on this axis)
zf 1024                               ! zero fill
red                                   ! reduced from complex to real
rft                                   ! real fourier transform
rev                                   ! reverse so correct frequencies
sto &col 0                            ! store column back into matrix
ty col = &col                         ! type that column is complete
next                                  ! get next column
end
```

Once Felix has been started, the first task is to build a matrix into which the data is to be placed.

The command (in Felix) for this is[1]

```
bld cosy 2 512 512 1
```

This creates a matrix files called cosy.mat which is 512x512 complex. Now the macro listed previously can be run which will read the data in and process it. The command for this (still in Felix) is

```
ex cosy.2.mac
```

where cosy.2.mac is the name of the file containing the macro[2]. The macro will ask you to input the matrix file name (cosy as named by the bld command) and the FID file name (felix as determined by the program when creating the output). Once this macro has been executed in Felix, the COSY spectrum is ready to be displayed. Commands to produce the contour plot on the screen are[3] as follows.

```
lvl 8e-4          ! set contour level low (~10^8 lower than real data)
cpn 1             ! both positive and negative contours
nl 5              ! five contour levels
cyc 2             ! even pen cycle to show positive and negatives
clm 2.5           ! contours increment 250 percent
cp                ! draw the contour plot to the screen
```

To get the axes to have the proper scaling one needs to use the rmx command[4]. The spectral width on both axes should be 333.33 Hz. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
hdv felix.hpgl    ! hardcopy device is file felix.hpgl
hpm 32            ! hardcopy plot mode is hpgl
hcp               ! produce the plot
```

This output file can be sent to a plotter which understands HPGL. As a last step for this example the felix output file, felix.hpgl, was converted to MIF format[5] in order to import the spectrum into

1. The matrix used is 512 x 512 complex. Currently the Felix documentation is contradictory in its statements on how to accomplish this. Experience says that either a 1 or a 2 at the end of the build statement produces a complex array, only a 0 produces a real array.
2. Felix may have trouble finding this macro file unless the “pre” command is used to specify which directory the macro files are located in, see the Felix documentation. In Felix, make sure that directory names end with a /. Keep in mind that Felix has problems (at least in UNIX) with capitol letters in filenames, best not to use and upper-case letters in your Felix related filenames. Finally, Felix has trouble with long path names. If your files are buried deep in some subdirectory use a link to shorten the path length down (see the UNIX man pages: man ln).
3. Early versions 1.0 of Felix seem to prefer that one draw 1D plots before 2D plots or it confuses itself on the plot limits. Apparently loading a row (loa 0 256) and drawing it (dr) then loading a column (loa 130 0) and drawing it prior to the contour plot does something to help Felix figure itself out. This seems not be a problem in later releases.
4. The command rmx is not fully documented in the Felix 1.0 manual. However there are scattered references to it throughout the examples. The command will query the user for needed parameters.
5. MIF stands for Maker Interchange Format used by FrameMaker. This conversion was done with a private filter function called Felix2mif and should reside (after GAMMA installation) in the GAMMA directory which contains the cosy example files.A similar program is supposed to be supplied by the manufacturers of FrameMaker in the near future. The program hpgltoeps that is supplied will perform a similar function and enables to spectrum to be imported into documents in encapsulated postscript format.

this FrameMaker document. The spectrum is nearly identical to that in the previous simulation.

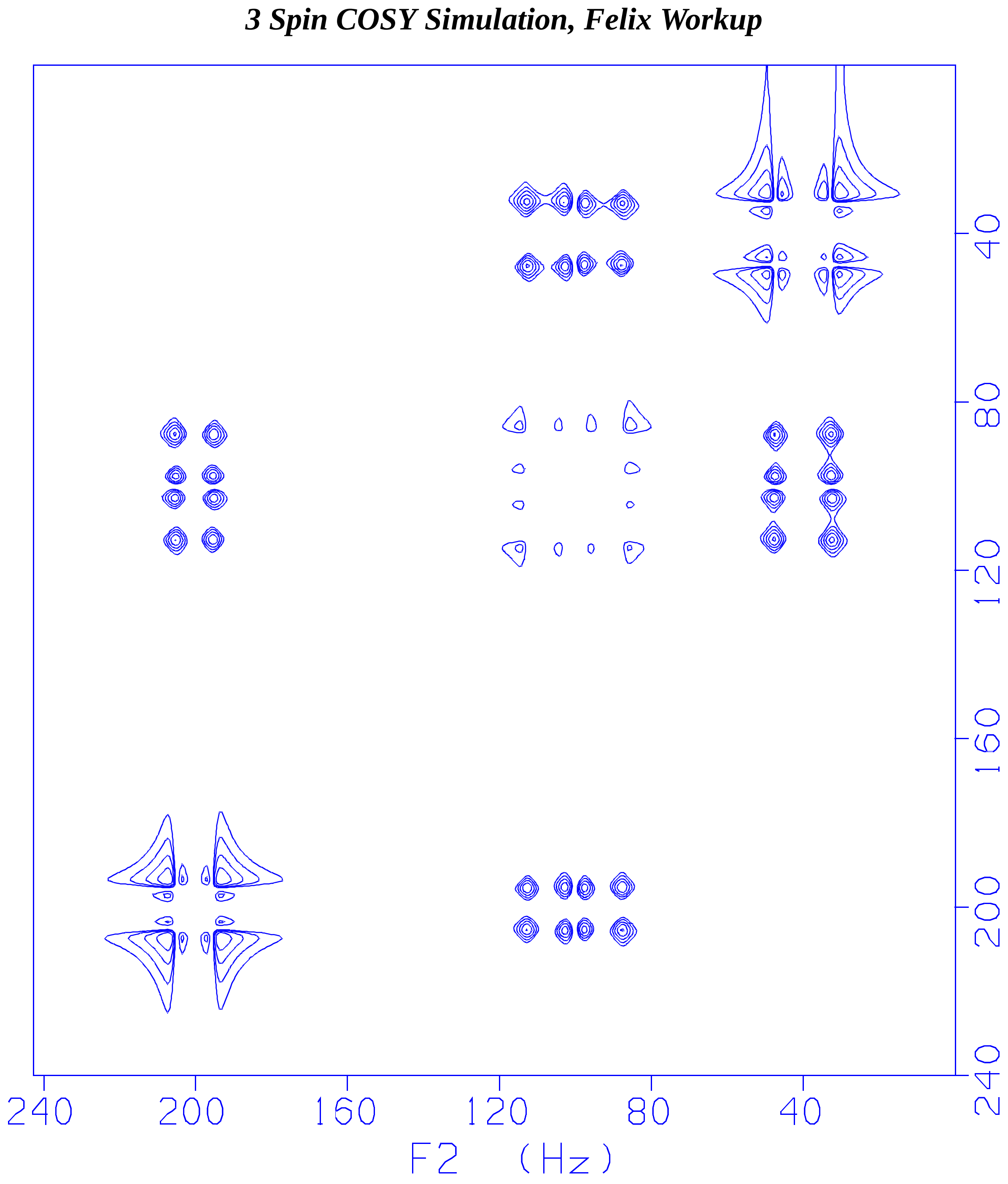


***Figure 0-5*** **Simulated COSY spectrum output from Felix, the result of COSY2.cc**

## 1.4 COSY with TPPI

### 1.4.1 Description

This example again performs a simple COSY simulation. In this instance, a TPPI[1] scheme is implemented in order to achieve quadrature detection along the $t_1$ axis. This is done by phase incrementing the initial pulse by 90 degrees on successive points along $t_1$. For clarity, the pulse sequence diagram is repeated with density matrices used to coincide with variable names in the program code.

***COSY Pulse Sequence***

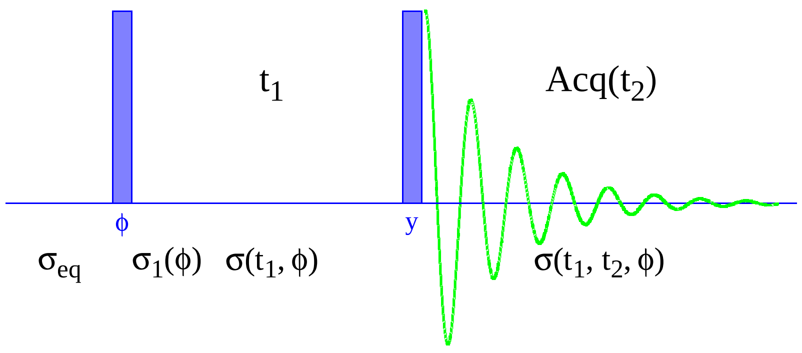


***Figure 0-6*** **A simple COSY pulse sequence. Density operators label key points in simulation of this exepriment.**

In this example, rather than incrementing the receiver phase (normal TPPI), for convenience the first pulse phase is incremented. No relaxation effects will be considered and the pulses taken as ideal. The resulting output file is produced in Felix format and subsequently processed with that program.

1. The TPPI detection scheme is discussed in Derome, *Modern NMR Techniques for Chemistry Research*, page 83.

## 1.4.2 Program

```
/* cosy3.cc ********************************************************************-*-c++-*-
**                                                                                        **
**                          Example program for the GAMMA Library                         **
**                                                                                        **
** This program simulates a basic COSY experiment. Phase cycling is                       **
** performed to achieve quadrature detection in t1.                                       **
**                                                                                        **
******************************************************************************************/
#include <gamma.h>
main ()
{
//                 DEFINE SYSTEM & NMR PARAMETERS

const int t1pts = 512;                              // points on t1 axis
const int t2pts = 512;                              // points on t2 axis
const double dt1 = 0.002;                           // t1 increment (SW 500 Hz, TPPI)
const double dt2 = 0.004;                           // t2 increment (SW 250 Hz)
spin_system sys;                                    // define the system, read in
sys.read("cosy.3.sys");                             // from disk
//                   SET UP NECESSARY VARIABLES

gen_op sigma1[4];                                   // density matrices, for TPPI
gen_op sigma;                                       // working density matrix
gen_op Upulse = Ixpuls_U(sys, 90.0);                // 90 x-pulse propagator
gen_op detect = Fm(sys);                            // F- for detection (0 phase)
gen_op H = Hcs(sys) + HJw(sys);                     // Hamiltonian, weak coupling
gen_op Udelay = Ie(sys,0);                          // t1 propagator (start = 1)
gen_op Udelay1 = prop(H, dt1);                      // t1 propagator increment
block_1D data(t2pts);                               // Storage for the FID
//                      APPLY PULSE SEQUENCE

gen_op sigma0 = sigma_eq(sys);                      // equilib. density mx
sigma1[0] = Ixypuls(sys,sigma0, 90.0, 0.0);         // Apply a (PI/2)x pulse
sigma1[1] = Ixypuls(sys,sigma0, 90.0, 90.0);        // Apply a (PI/2)y pulse
sigma1[2] = Ixypuls(sys,sigma0, 90.0, 180.0);       // Apply a (PI/2)-x pulse
sigma1[3] = Ixypuls(sys,sigma0, 90.0, 270.0);       // Apply a (PI/2)-y pulse
File fp;
fp.open("felix.dat", io_writeonly, a_create);       // Create file,open
for(int t1=0; t1<t1pts; t1++)
  {
  sigma = evolve(sigma1[t1%4], Udelay);             // TPPI Cycle, 1st pulse
  evolve_ip(sigma, Upulse);                         // apply 2nd 90 x-pulse
  FID(sigma, detect, H, dt2, t2pts, data);          // acquisition
  Felix(fp, data);                                  // write block
  Udelay *= Udelay1;                                // increment t1
  }
fp.close();
}
```

## 1.4.3 Discussion

**Define System and NMR Parameters:** The first four lines specify the dwell times and number of points to use on the $t_1$ & $t_2$ axes. Note that the spectral width is set two twice that actually desired along $t_1$ due to use of TPPI. The spin system used was a slight variation of the three spin homonuclear system used in the first two examples. It is read in from disk file cosy3.sys (see next section).

**Set up the Necessary Variables:** Initially four density matrices are set up for the four phases to be used in TPPI mode following the first 90 pulse. Another working density matrix is also allocated. Next, a pulse propagator is created for the 90x pulse and will be used inside the loop over $t_1$ increments. The detector is set to $F_- = F_x - iF_y$ so as to produce a complex data set along the t2 axis. The Hamiltonian is set for weak scalar coupling (under the secular approximation). Two more propagators are now set up. The first is a working propagator which will propagate the density matrix through the total $t_1$ time. This is initialized to the identity matrix. The 2nd propagator is that for the evolution over a single $t_1$ increment. Following this, a data block is allocated for the acquisitions.

**Apply the Pulse Sequence:** The program sets the initial density matrix, $\sigma_0$, to the equilibrium density matrix. A ninety degree pulse is now applied, four times at different phases to produce four $\sigma_1$ matrices corresponding to the four phases. These are calculated outside the loop over $t_1$ times. Just before the loop a file is opened called "felix.dat" for the spectral output. The looping begins over all $t_1$ points. The first loop step is to evolve the appropriate $\sigma_1$ for the $t_1$ time to produce $\sigma(t_1, \phi)$. In this instance, modulus arithmetic is utilized to convert the $t_1$ increment into an integer 0,1,2, or 3 corresponding to the phases 0, 90, 180, and 270, The second 90 pulse of the sequence is then applied and the xy-magnetization determined. This is an acquisition step which fills the data block data up with t2pts FID points. The next step in the loop is to write this block out to the file “felix.dat” in Felix format. At the end of the loop, the propagator for the $t_1$ delay is incremented to the total delay time in the loop. The last line of the program simply closes the output file.

## 1.4.4 Spin System

The spin system file, cosy3.sys, is below. It is frequency shifted from the one used previously.

```
SysName  (2) : COSY_TPPI  - Name of the Spin System
NSpins   (0) : 3          - Number of Spins in the System
Iso(0)   (2) : 1H         - Spin Isotope Type
Iso(1)   (2) : 1H         - Spin Isotope Type
Iso(2)   (2) : 1H         - Spin Isotope Type
v(0)     (1) : 200.0      - Chemical Shifts in Hz
v(1)     (1) : 100.0      - Chemical Shifts in Hz
v(2)     (1) : 40.0       - Chemical Shifts in Hz
J(0,1)   (1) : 10.0       - Coupling Constants in Hz
J(0,2)   (1) : 0.0        - Coupling Constants in Hz
J(1,2)   (1) : 15.0       - Coupling Constants in Hz
Omega    (1) : 400        - Spectrometer Frequency in MHz (1H based)
```

## 1.4.5 Results

Workup of the simulated data was performed in Felix. A text editor was used to create the following Felix macro file, called cosy3.mac (the comments off to the side are not part of the macro) -

```
cl                                  ! insure any data files are closed
cmx                                 ! insure any matrix files are closed
get ‘Enter matrix name: ‘ mname     ! ask user for matrix
mat &mname write                    ! open matrix with write ability
get ‘Enter FID name: ‘ dname        ! ask user for fid file name
lb 4                                ! set line broadening
ph0 90                              ! set zero order phase correction
for row 1 512                       ! loop through all 512 rows
re &dname                           ! read FID from file
em                                  ! apodize
ft                                  ! fourier transform
ph                                  ! apply phase correction
rev                                 ! reverse spectrum so frequencies proper
si 512                              ! reduce the size back to 512
sto 0 &row                          ! store block in matrix row
ty row = &row                       ! type row is complete
next                                ! get next row
for col 1 512                       ! begin column processing
loa &col 0                          ! retrieve column from matrix
em                                  ! apodize
zf 1024                             ! zero fill
rft                                 ! real fourier transform
red                                 ! reduced from complex to real
rev                                 ! reverse so correct frequencies
sto &col 0                          ! store column back into matrix
ty col = &col                       ! type that column is complete
```

```
        next                          ! get next column
        end
```

Once Felix has been started, the first task is to build a matrix into which the data is to be placed. The command for this is (in Felix)[1]

```
        bld cosy 2 512 512 0
```

This creates a matrix files called cosy.mat which is 512x512 real. Now the macro listed previously can be run which will read the data in and process it. The command for this (still in Felix) is

```
        ex cosy3
```

where cosy3.mac is the name of the file containing the macro[2]. Once this macro has been executed in Felix, the COSY spectrum is ready to be displayed. Commands to produce the contour plot on the screen are[3] -

```
        lvl 1.3e-4                    ! set contour level very low (10**8 lower than exptl.)
        cpn 1                         ! both positive and negative contours
        nl 5                          ! five contour levels
        cyc 2                         ! even pen cycle to show positive and negatives
        cp                            ! draw the contour plot to the screen
```

To get the axes to have the proper scaling one needs to use the rmx command[4]. The spectral widths on both axes should be 250 Hz and the 256th point set to 120 Hz. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
        hdv felix.hpgl                ! hardcopy device is file felix.hpgl
        hpm 32                        ! hardcopy plot mode is hpgl
        hcp                           ! produce the plot
```

As a last step, the felix output file, felix.hpgl was converted to mif format in order to import the spectrum into this document. The spectrum is nearly identical to that in the previous simulation.

1. The matrix used is 512 x 512 real. Currently the Felix documentation is contradictory in its statements on how to accomplish this. Experience says that a 0 at the end of the build statement produces a real array, a 1 or 2 produces a complex array.
2. Felix may have trouble finding this macro file unless the "pre" command is used to specify which directory the macro files are located in. Keep in mind that Felix has problems (at least in UNIX) with capitol letters in filenames and that directory names need to end with a /.
3. Early versions 1.0 of Felix seem to prefer that one draw 1D plots before 2D plots or it confuses itself on the plot limits. Apparently loading a row (loa 0 256) and drawing it (dr) then loading a column (loa 130 0) and drawing it prior to the contour plot does something to help Felix figure itself out. This seems not be a problem in later releases.
4. The command rmx is not fully documented in the Felix 1.0 manual. However there are scattered references to it throughout the examples. The command will query the user for needed parameters.

***3 Spin COSY Simulation, TPPI, Felix Workup***

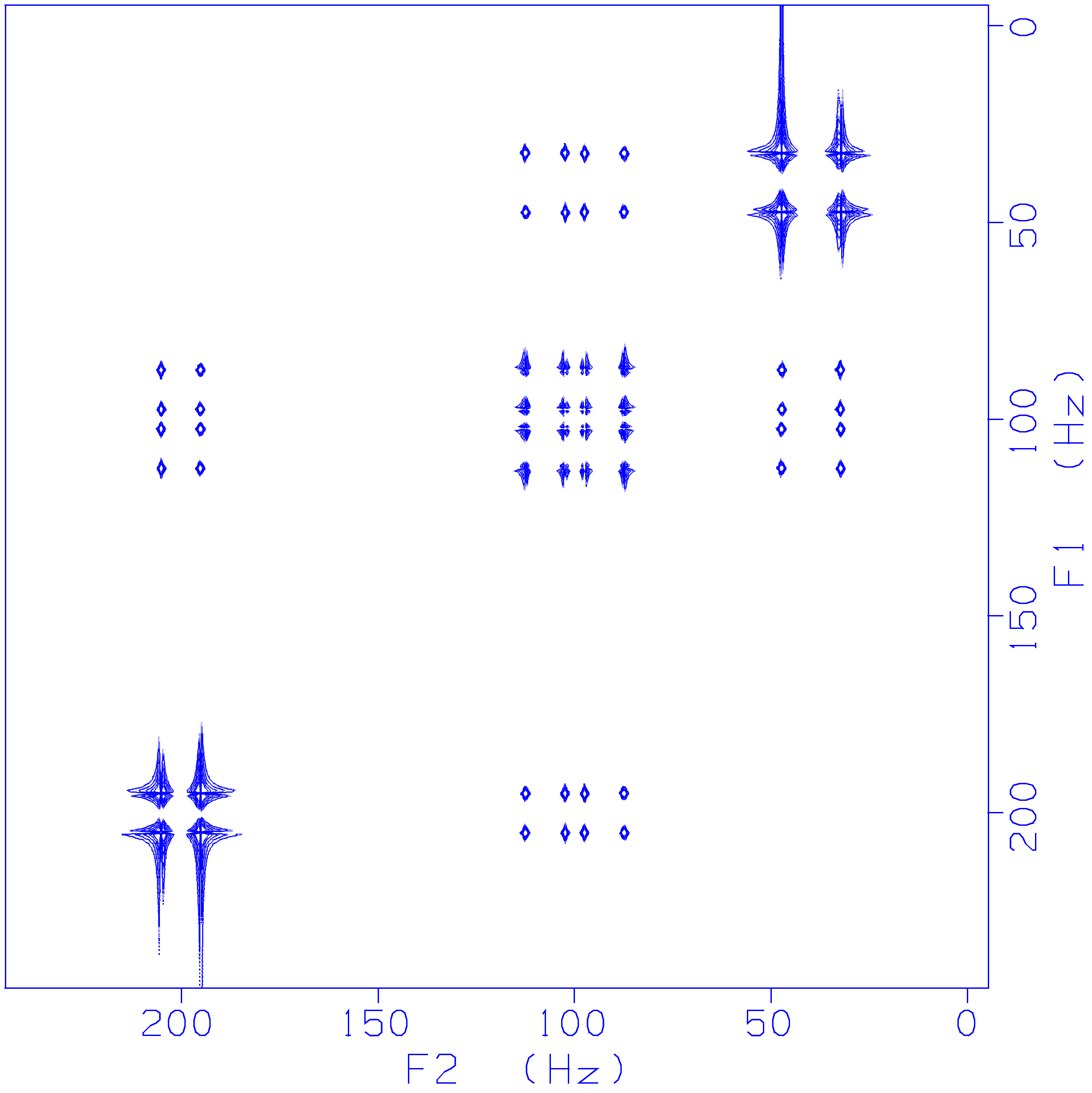


*Figure 0-7* **Simulated COSY spectrum from ccosy3.cc. The system used in this simulation had all frequencies shifted by 120 Hz, effectively placing the carrier in the middle of the spectrum. This value was reinserted in the workup so that the spectrum has the same frequencies as in previous examples.**

## 1.5 COSY with RuSH Method

### 1.5.1 Description

For our final example of a simple COSY simulation we use the RuSH method[1] in order to achieve quadrature detection along the $t_1$ axis. To accomplish this we essentially perform two simultaneous simulations (experiments), the second being a phase shifted version of the first. Data from the first experiment is then considered during $t_1$ processing as the “real” data and that of the second experiment as the “imaginary” data. The pulse sequence diagram to be utilized is shown below with density matrices labeled to coincide with the variable names used in the program code.

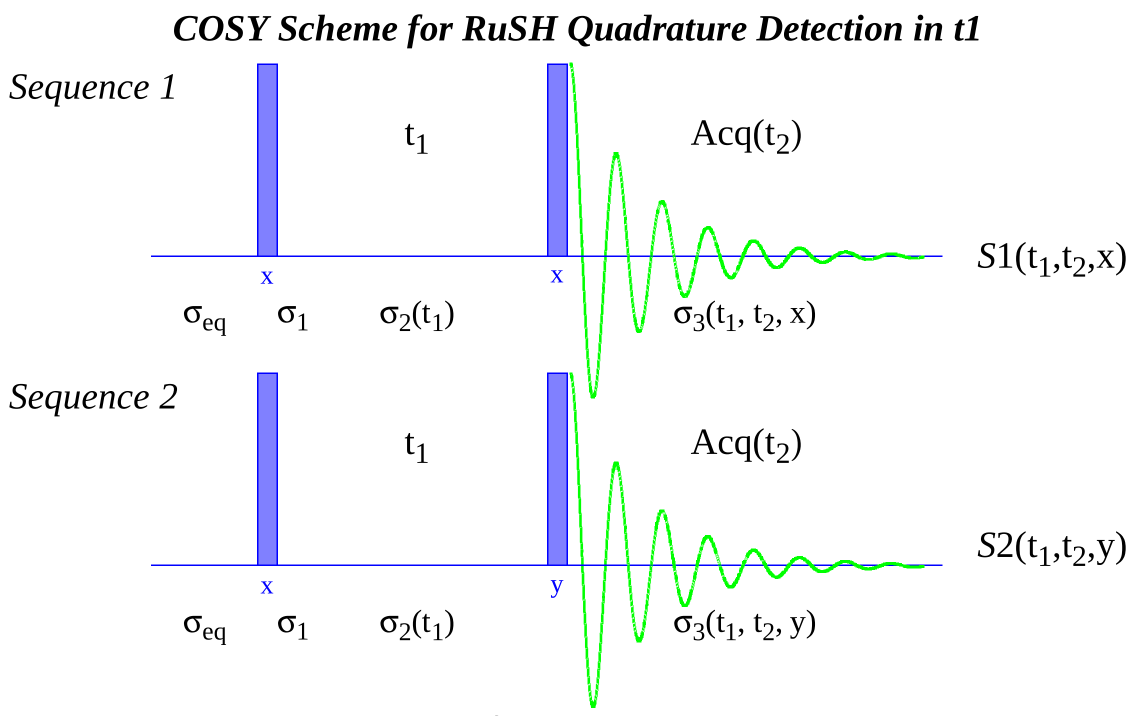


***Figure 0-8*** **Two COSY pulse sequences for RUSH. Density operators label key points in simulation.**

In this simulation two data sets (labeled S1 and S2 in the figure) are generated. Each contain different phase information and they need to be combined for the quadrature detection in $t_1$. They are recombined during spectral workup in the following fashion.

COSY Data Workup for RuSH Quadrature Detection

$$\begin{tikzcd}
S1(t_1,t_2,x) \arrow[r, "FFT(t_2)"] & S1(t_1,f_2,x) \arrow[r, "Re"] & RS1(t_1,t_2,x) \arrow[dr, "x"] & & \\
 & & & S(t_1,f_2,x+iy) \arrow[r, "FFT(t_1)"] & S1(f_1,f_2) \\
S2(t_1,t_2,y) \arrow[r, "FFT(t_2)"] & S2(t_1,f_2,y) \arrow[r, "Re"] & RS2(t_1,t_2,y) \arrow[ur, "iy"'] & &
\end{tikzcd}$$

1. The RuSH method is discussed in Derome, *Modern NMR Techniques for Chemistry Research*, page 201.

### 1.5.2 Program

```
/* cosyrush.cc *****************************************************************-*-c++-*-
**                                                                              **
**                     Example program for the GAMMA Project                    **
**                                                                              **
** The Program reads a spin system (number of spins, chemical shifts and        **
** J-coupling constants) from a file and simulates a normal COSY experiment     **
** using RuSH for quadrature detection in t1                                    **
**                                                                              **
** NOTE: Set to work only on HOMONUCLEAR systems                                **
**                                                                              **
********************************************************************************* */

#include <gamma.h>
//                        Define Constants

const int t1pts = 512;                              // Number of t1 points
 const int t2pts = 512;                             // Number of t2 points
//                        Begin Program

main (int argc, char* argv[])

{
 cout << "\Homonuclear COSY Simulation with RuSH Mode\n";
//                        Read in the Spin System

String filename;                                    // Name of spin system file
 query_parameter(argc, argv, 1,                     // Get filename from command
        "\nSpin system filename? ", filename);      // line or ask for it
 spin_system sys;                                   // Declare spin system sys
 sys.read(filename);                                // Read system from filename
//                   Set Offsets and Spectral Widths

double offset = sys.center();                       // Find approx. spectrum center
 sys.offsetShifts(offset);                          // Offset shifts so centered
 double NyqF = sys.Nyquist(0, 1.4);                 // Approximate Nyquist frequency
 double t2dt = 1.0/(2.0*NyqF);                      // t2 time increment
 double t1dt = t2dt;                                // t1 time increment
//                        Set Up Hamiltonian

 char J;
 query_parameter(argc, argv, 2,                     // Weak or strong coupling
          "\nWeak or strong coupling (w/s)?", J);
 gen_op H;                                          // Set Hamiltonian for
 if (J == 'w')                                      // strong or weak coupling
 H = Hcs(sys) + HJw(sys);
 else
 H = Hcs(sys) + HJ(sys);
 gen_op Upx = Ixypuls_U(sys,0.0,90.0);              // Propagator for x pulse
 gen_op Upy = Ixypuls_U(sys,90.0,90.0);             // Propagator for y pulse
 gen_op Ud1 = prop(H, t1dt);                        // Propagator t1 delay increment
 gen_op D = Fm(sys);                                // Detector to F-
 gen_op sigma0, sigma1, sigma2, sigma3;             // Set up density matrices
 block_1D t2BLK(t2pts);                             // Set 1D block for output
//                   Pulse Sequence and I/O Setup

File cosyRe, cosyIm;                                // Declare and open two files
 cosyRe.open("cosyRuSH.Redat",io_writeonly, a_create);
 cosyIm.open("cosyRuSH.Imdat",io_writeonly, a_create);
//                      Apply Pulse Sequence

gen_op sigma0 = sigma_eq(sys);                      // Set density matrix equilibrium
 gen_op sigma1 = evolve(sigma0, Upx);               // Apply first (PI/2)x pulse
 gen_op sigma2 = sigma1;                            // Initial sigma2 (t1 = 0)
 for (int t1=0; t1<t1pts; t1++)                     // Loop over all t1 increments
      {
      sigma3 = evolve(sigma2, Upx);                 // Apply second (PI/2)x pulse
      FID(sigma3,D,H,t2dt,t2pts,t2BLK);             //Acquire data for reals
      Felix(cosyRe, t2BLK);                         // Output block for reals: Felix
      sigma3 = evolve(sigma2, Upy);                 // Apply second (PI/2)y pulse
      FID(sigma3,D,H,t2dt,t2pts,t2BLK);             //Acquire data for imaginaries
      Felix(cosyIm, t2BLK);                         // Output block for imags: Felix
      evolve_ip(sigma2, Ud1);                       //evolution to next t1
      }
 cosyRe.close();                                    // Close files
 cosyIm.close();
}
```

### 1.5.3 Discussion

**Define System and NMR Parameters:** The first two line set the number of points to simulation along each dimension. These can be altered before compilation to suit individual needs. The program begins at the main statement, in this case main itself has arguments which allow the program to take use commands given on the command line. Following a brief output about the type of simulation, the spin system is read in from an external file. First a string variable “filename” is declared. The filename is set to be the first parameter given on the command line or, if none is given there, the user is asked for the filename. Having the spin system filename, a spin system variable “sys” is declared and read in from the file. Since the program is “automated” it finds an appropriate frequency offset (for better digital resolution) and shifts the spin system by this amount. An approximate Nyquist frequency is obtained from the function “Nyquist” and this is used to set the dwell times on both axes, “t1dt” and “t2dt” respectively.

**Set up the Necessary Operators:** It is left to the user whether the simulation should be done with strong or weak coupling. Again, this will be set from the second parameter on the command line or, if not given on the command line, the user will be prompted to input which is preferred. A general operator H is declared and set to be the sum of the isotropic shift Hamiltonian plus the chosen isotropic scalar coupling Hamiltonian. After this, three propagators (general operators) are created. These are Upx, Upy, and Ud1 for the 90x pulse, the 90y pulse, and the $t_1$ time increment - all of these are constant throughout the simulation and need to be computed only once. Also computed once is the general operator D used for the detection. The detector, D, is set to $F_- = F_x - iF_y$ so as to produce a complex data set along the t2 axis. We then declare the four density matrices (corresponding to the pulse sequence diagram) and a data block “t2BLK” in which to collect the FID’s.

**Apply the Pulse Sequence:** The program sets the initial density matrix, $\sigma_0$, to the equilibrium density matrix. A ninety degree pulse is then applied along the x-axis to produce $\sigma_1$, and then the first $\sigma_2$, $\sigma_2(t_1=0) = \sigma_1$, is set. Then begins the looping over all $t_1$ points desired. The density matrix $\sigma_3(t_1,x)$ is generated from applying a 90x pulse to $\sigma_2(t_1)$ and in turn $\sigma_3(t_1,x)$ is propagated in $t_2$ to produce $\sigma_3(t_1, t_2, x)$ and a corresponding FID which is output to the file cosyRe, “recosy.dat” in Felix compatible format. Then, the same $\sigma_2(t_1)$ is used to produce another $\sigma_3$ but this time with the application of a 90y pulse. Again $\sigma_3(t_1,y)$ is used to generate $\sigma_3(t_1, t_2, y)$ and a corresponding FID which is now output to the file cosyIm, “imcosy.dat”. The loop is then repeated for each $t_1$ but, rather than produce $\sigma_2(t_1)$ from $\sigma_1$ by evolving it for time $t_1$ we compute $\sigma_2(t_1)$ from $\sigma_2(t_1-\Delta t_1)$ by evolving it for time $\Delta t_1$. This is performed in the last step of the loop. The last two lines of the program close the two output files.

### 1.5.4 Felix Processing

The current version of Felix comes with a manual loaded with examples of macros for performing data processing. Unfortunately, most of these simply do not work at all[1]. In this particular case, unlike processing a data set acquired using TPPI, we must take a complex Fourier transform along the $t_1$ axis. Since we also want to produce a contour plot of the processed real data we would prefer to maintain all data in a Felix real matrix. The Felix manual is very misleading and vague concerning this type of treatment so here we cover the required processing in fine detail. The following figure diagrams how the two data sets *S1* and *S2* are initially worked up in the t2/F2 dimension.

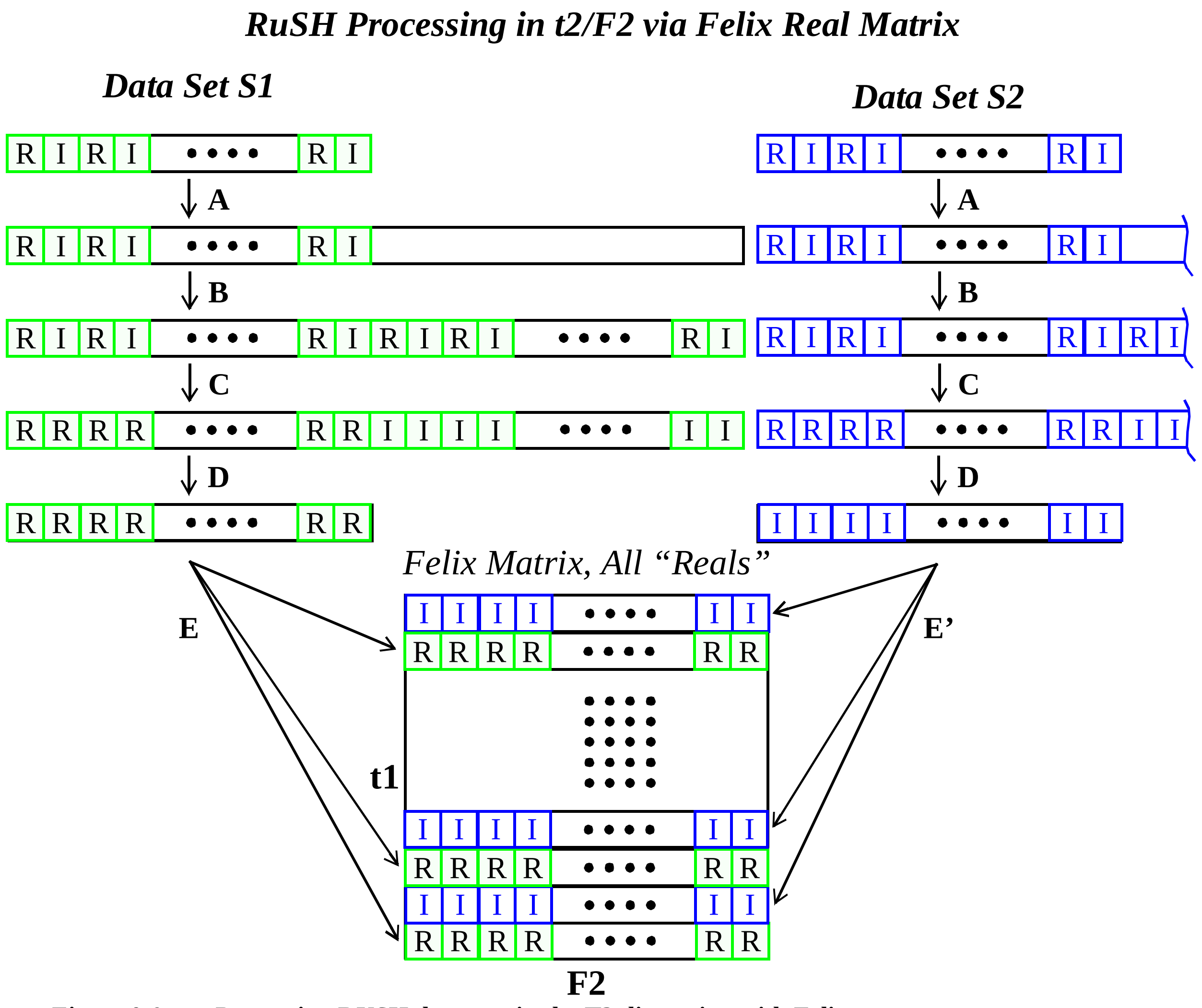


*Figure 0-9* **Processing RUSH data sets in the T2 dimension with Felix.**

1. The manual currently being referred to is dated October 1990 and is for Felix Version 1.1. This manual exemplifies how extremely difficult it is to keep user documentation for a program current (hopefully GAMMA docs. aren't too similar in this point). We therefore tend to always include the Felix macros used in processing our simulated data in discussing these examples. From past experience, there is absolutely no guarantee that any of our listed macros will work in future Felix versions!

As each FID block is read in from the respective data file it is initially (**A**) zero filled to the Felix matrix dimension and then (**B**) Fourier transformed. The block is now complex frequency data and the reals and imaginaries contain redundant information since they differ only by a phase change. The block is also too large to fit into a row of the Felix matrix- the block dimension is correct but the matrix only has room for real data, not complex data. The block will be cut in half and fit correctly but must first be manipulated to isolate the useful data. In the next step (**C**), the reals are completely separated from the imaginaries. In the following step (**D**) the imaginaries are thrown out, they contain no new information, and the block not contains only real frequency information. The final step (**E**) puts the real frequency data into the Felix matrix row. Here, the blocks from the two data sets must be handled slightly differently (**E** & **E'**). We are storing all data in a real Felix matrix, yet we wish to perform a complex Fourier transform along $t_1$. Since Felix always believes complex points are stored in (real, imaginary) pairs we must store or blocks in "real" block, "imaginary" block pairs. Since our "real" data comes from the *S1* data set and the "imaginary" data comes from the *S2* data set we must store the blocks in an interleaved order.

## *RuSH Processing in t1/F1 via Felix Real Matrix*

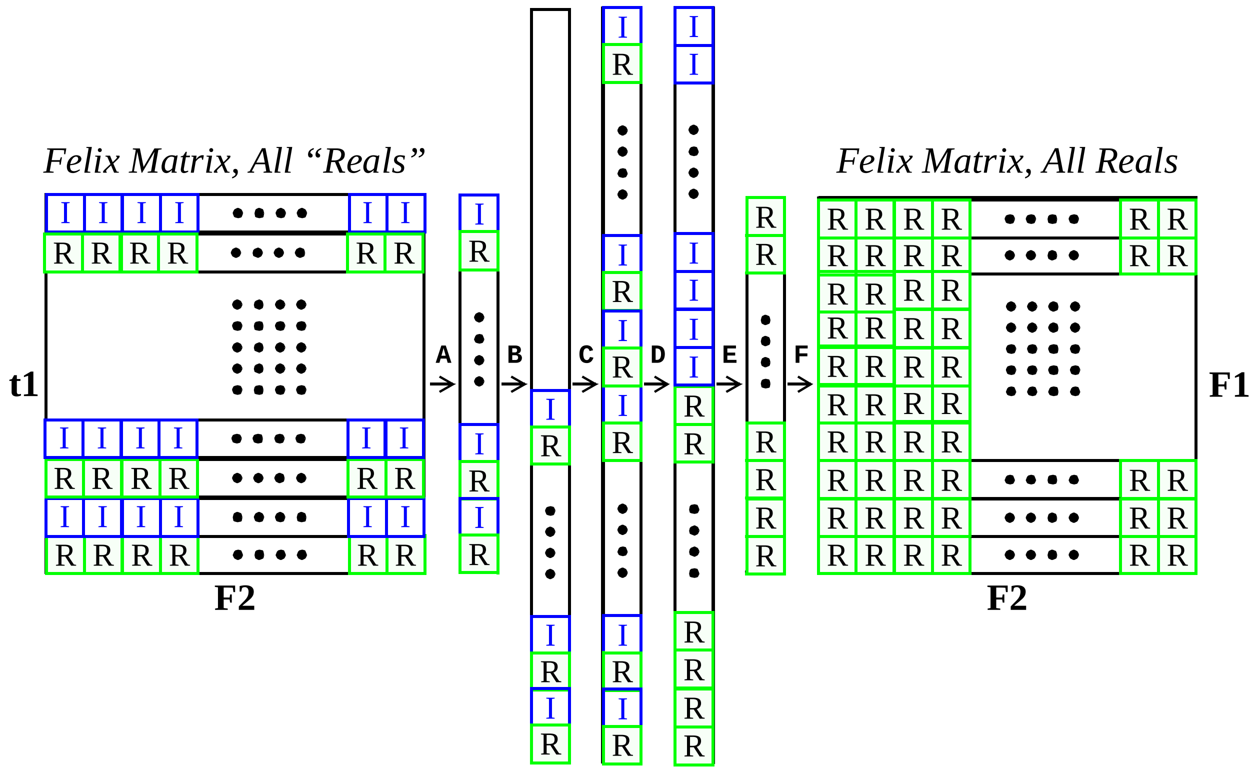


***Figure 0-10*** **Processing RUSH data sets in the T1 dimension with Felix.**

We now discuss how Felix processes the data along the t1/F1 dimension. Each column is successively pulled out of the Felix real matrix (**A**). Since the matrix is real, the data will be real to Felix even though we have constructed the columns so they are in the Felix complex column format. The program must at this point be told the data is complex after which the data size will register as 1/2 the value when removed from the matrix. The now complex data is zero filled to the matrix column dimension (**B**) and Fourier transformed (**C**). As in the t2/F2 processing, we now have complex data which is too large to fit into the matrix but we only want the reals. So, we sort the reals and imaginaries (**D**) then throw away the imaginary part (**E**). Felix is told that the data is again real and it is then stored back into the matrix (**F**).

A conceptually simpler Felix workup can be performed using a complex array. The trade-off in simplicity is that the disk space used by the matrix twice that used in the macro manipulating the data in a real matrix. The diagram below shows how processing is done with the complex matrix.

***RuSH Processing in t1/F1 via Felix Complex Matrix***

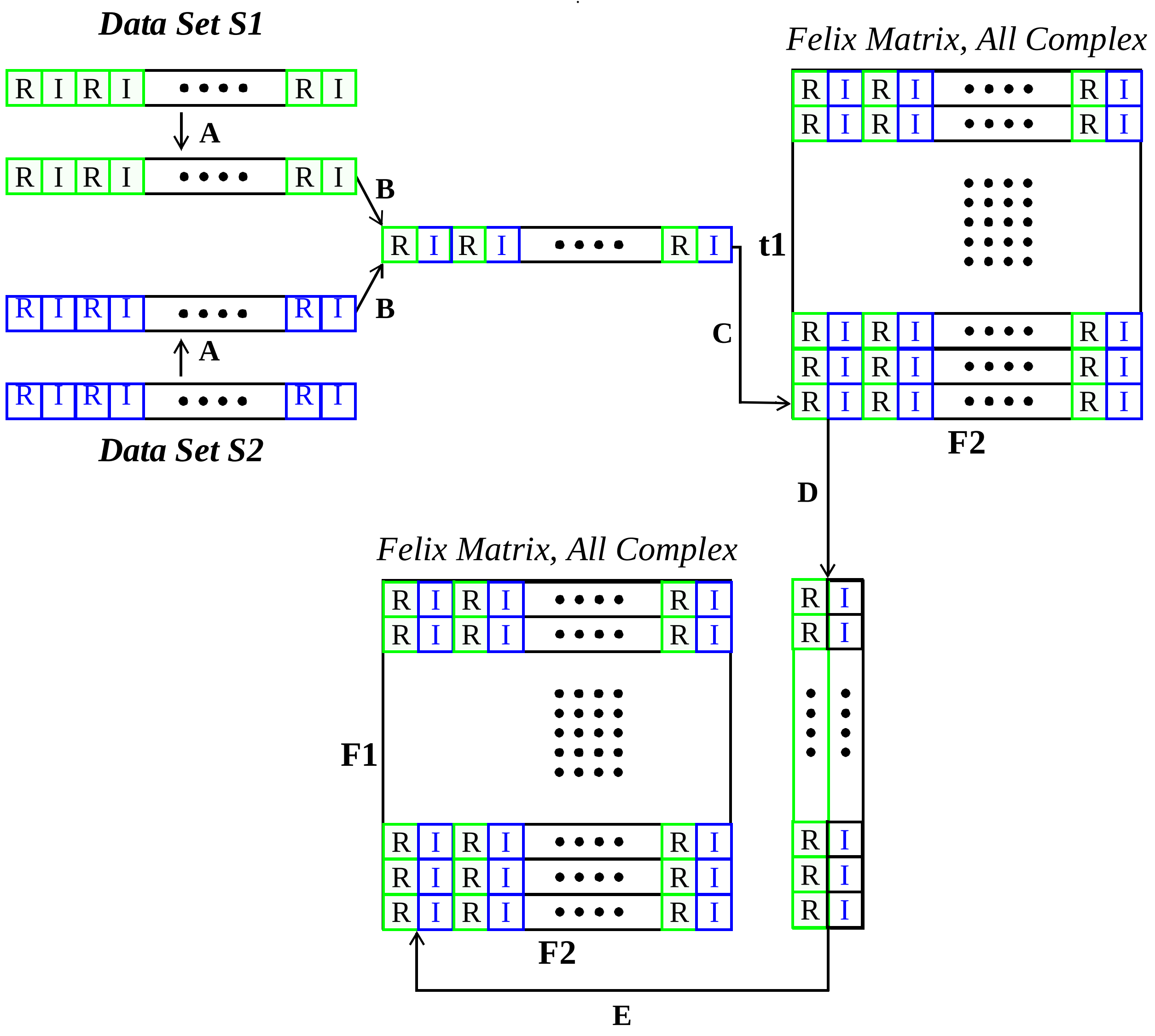


*Figure 0-11* **Processing RUSH data sets in the T1 dimension with Felix.**

Rows are successively read from each data set, *S1* and *S2* and Fourier transformed along the $t_2$ dimension (**A**). The real frequency data from *S1* is blended with the imaginary frequency data from *S2* (**B**) to form a new complex data set having the appropriate phase information for quadrature processing in $t_1$. This is stored into the Felix complex matrix row (**C**). Afterwards, successive complex columns are read from the matrix (**D**), Fourier transformed and restored into the complex array (**E**).

### 1.5.5 Spin System

The spin system file, cosy.3.sys, is frequency shifted from the one used in the previous simulation. It is reproduced below.

```
SysName  (2) : COSYRuSH  - Name of the Spin System
NSpins   (0) : 3         - Number of Spins in the System
Iso(0)   (2) : 1H        - Spin Isotope Type
Iso(1)   (2) : 1H        - Spin Isotope Type
Iso(2)   (2) : 1H        - Spin Isotope Type
v(0)     (1) : 200.0     - Chemical Shifts in Hz
v(1)     (1) : 100.0     - Chemical Shifts in Hz
v(2)     (1) : 40.0      - Chemical Shifts in Hz
J(0,1)   (1) : 10.0      - Coupling Constants in Hz
J(0,2)   (1) : 0.0       - Coupling Constants in Hz
J(1,2)   (1) : 15.0      - Coupling Constants in Hz
Omega    (1) : 400       - Spectrometer Frequency in MHz (1H based)
```

### 1.5.6 Workup

We provide two Felix macros for data processing. As discussed in a previous section, we wish to use real matrices when working up the data and the corresponding macro is provided. A second macro is also given, one in which the data is stored in a complex Felix matrix. The former saves on disk space while the latter is perhaps more straight forward.

**COSY RuSH Macro Using Felix Real Matrix**

```
def redat cosyre                          ! set up a variable for the “real” data file
def imdat cosyim                          ! set up a variable for the “imag” data file
def matfile cosy                          ! set up a variable for the real Felix matrix file
def t1max 512
lb 10
ph0 0
ty Opening matrix file “&matfile “ .....  ! report of macro progress
cmx                                       ! close any open matrix
cl                                        ! close any open files
mat &matfile write                        ! open the matrix file for writing
ty Using real data file “&redat “ ......  ! another status report
ty Complex-FT of t2 into real F2 ........
for row 1 &t1max                          ! begin looping through the t1 values
re &redat                                 ! read in a complex FID from the “real” data
zf 1024
em                                        ! apodize the FID
ft                                        ! Fourier transform the complex data
ph
rev                                       ! reverse the frequencies for plotting
sep                                       ! split up all real/imaginary pairs
dat 0                                     ! call the data real, now twice a big
si 1024                                   ! use only the first half, all the true reals
sto 0 &nrow                               ! store real row into real matrix - 1,3,5,7,9,...
ty reals row &row - matrix row &nrow
eva nrow (&nrow+2)                        ! increment matrix storage row count by two
next                                      ! back for next row of “real” input data
ty Using imaginary data file “&imdat “ .  !
ty Complex-FT of t2 into imaginary F2 ... !
def nrow 2                                ! reset the matrix storage row to be initially 2
for row 1 &t1max
re &imdat                                 ! read in a complex FID from the “imag” data
zf 1024
em                                        ! apodize the FID
ft                                        ! Fourier transform the complex data
rev                                       ! reverse the frequencies for plotting
exc
sep                                       ! split up all real/imaginary pairs
dat 0                                     ! call the data real, now twice a big
si 1024
sto 0 &nrow                               ! store real row into real matrix - 2,4,6,8,...
ty imags row &row - matrix row &nrow
eva nrow (&nrow+2)
next                                      ! back for next row of “imag” input data
ty Complex-FT of t1 into F1 ......
```

```
for col 1 1024
loa &col 0                            ! load column of real data from real matrix
dat 1                                 ! but it is actually complex data, half the size
si 512
zf 1024
em                                    ! apodize the FID
ft                                    ! Fourier transform the complex data
ph
rev                                   ! reverse the frequencies for plotting
sep                                   ! split up all real/imaginary pairs
dat 0                                 ! again call the data real, now twice a big
si 1024
sto &col 0                            ! store the real data back into the real matrix
ty col &col
next
ty Setting Up Contour Plot........
cmx
mat &matfile read
lim 1 1 1024
lim 2 1 1024
lvl 1e-4
cpn 1
pen 1
nl 4
cyc 2
rmx 1 400 239 3 512 0 F2
rmx 2 400 238 3 512 0 F1
ty Plotting contours ..........
cp
ty Processing finished ........
end
```

## COSY RuSH Macro Using Felix Complex Matrix

```
def redat cosyre                      ! set up a variable for the “real” data file
def imdat cosyim                      ! set up a variable for the “imag” data file
def matfile cosy                      ! set variable for the complex Felix matrix file
def t1max 512
def t2max 512
def total 512
lb 10
ph0 0
ty Opening matrix file “&matfile “ .....   ! report of macro progress
cmx                                   ! close any open matrix
cl                                    ! close any open files
mat &matfile write                    ! open the matrix file for writing
ty Using real data file “&redat “ ......   ! another status report
ty Complex-FT of t2 into real F2 ........
for row 1 &t1max                      ! begin looping through the t1 values
re &redat                             ! read in an FID from the “real” data
em                                    ! apodize the FID
ft                                    ! Fourier transform the complex data
zi                                    ! zero the imaginaries, they’ll be replaced
sto 0 &row                            ! store the data into the complex matrix
ty reals row &row                     ! tell which row is being processed
next                                  ! go to the next row
```

```
ty Using imaginary data file “&imdat “ .     ! another status report
ty Complex-FT of t2 into imaginary F2 ...
for row 1 &t1max                             ! begin looping through the t1 values
loa 0 &row                                   ! read corresponding matrix row or “reals”
stb 1                                        ! store the complex row with reals in a buffer
re &imdat                                    ! read corresponding row of “imags” data
si &t2max
em                                           ! apodize the FID
ft                                           ! Fourier transform the complex data
zr                                           ! zero the reals this time
adb 1                                        ! add these imaginaries to the reals in buffer
ldb 1                                        ! get the new complex row from the buffer
rev                                          ! reverse the frequency order for plotting
sto 0 &row                                   ! store the row back into the matrix
ty imags row &row                            ! tell which row is being processed
next                                         ! proceed to next row
ty Complex-FT of t1 into F1 ......           ! report that the columns are ready
for col 1 &total                             ! begin looping through the columns
loa &col 0                                   ! read in a complex column
si &t1max
em                                           ! apodize the FID
ft                                           ! Fourier transform the complex data
ph 0
rev                                          ! reverse the frequency order for plotting
sto &col 0                                   ! store the complex column back in the matrix
ty col &col                                  ! report on the column progress
next                                         ! go get the next column
ty Setting Up Contour Plot........           ! status report that plotting is ready
cmx                                          ! first close the matrix
mat &matfile read                            ! reopen the matrix, now read only for safety!
lim 1 1 512                                  ! set the plotting limits
lim 2 1 512
lvl 1e-4                                     ! set the initial contour intensity
cpn 1                                        ! contour both positive and negative peaks
pen 1                                        ! begin with pen 1 on first contour
nl 4                                         ! plot four contour levels
cyc 2                                        ! cycle the pen, + one color, - another color
rmx 1 300 1596 3 256 5.3 F2                  ! set the axes up
rmx 2 300 1596 3 256 5.3 F1
ty Plotting contours ..........              ! status report that plotting is starting
cp                                           ! draw a contour plot
ty Processing finished ........              ! status report that the macro is through
end                                          ! end the macro
```

## 1.5.7 Results

### COSY RuSH Macro Using Felix Real Matrix

Once Felix has been started, the first task is to build a real matrix into which the data is to be placed. The command for this is (in Felix)[1]

```
bld cosy 2 1024 1024 0
```

This creates a real matrix file called cosy.mat which is 1Kx1K. Now the macro listed previously can be run which will read the data in and process it. The command for this (still in Felix) is

```
ex recosy4
```

where recosy4.mac is the name of the file containing the macro[2]. Once this macro has been executed in Felix, the COSY spectrum will automatically be displayed. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
hdv felix.hpgl          ! hardcopy device is file felix.hpgl
hpm 32                  ! hardcopy plot mode is hpgl
hcp                     ! produce the plot
```

### COSY RuSH Macro Using Felix Complex Matrix

Once Felix has been started, the first task is to build a complex matrix into which the data is to be placed. The command for this is (in Felix)[3]

```
bld cosy 2 512 512 1
```

This creates a complex matrix file called cosy.mat which is 512x512. Now the macro listed previously can be run which will read the data in and process it. The command for this (still in Felix) is

```
ex cmcosy4
```

where recosy4.mac is the name of the file containing the macro[4]. Once this macro has been executed in Felix, the COSY spectrum will automatically be displayed. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
hdv felix.hpgl          ! hardcopy device is file felix.hpgl
hpm 32                  ! hardcopy plot mode is hpgl
hcp                     ! produce the plot
```

1. Currently the Felix documentation is contradictory in its use of bld. Experience says that a 0 at the end of the build statement produces a real array, a 1 or 2 produces a complex array.
2. Felix may have trouble finding this macro file unless the “pre” command is used to specify which directory the macro files are located in. Keep in mind that Felix has problems (at least in UNIX) with capitol letters in filenames and that directory names need to end with a /.
3. See footnote 1.
4. See footnote 4.

## 1.6 COSY with real pulses

### 1.6.1 Description

This example again performs a simple COSY simulation. In this instance, a TPPI[1] scheme is implemented in order to achieve quadrature detection along the $t_1$ axis. This is done by phase incrementing the initial pulse by 90 degrees on successive points along $t_1$. For clarity, the pulse sequence diagram is repeated with density matrices used to coincide with variable names in the program code.

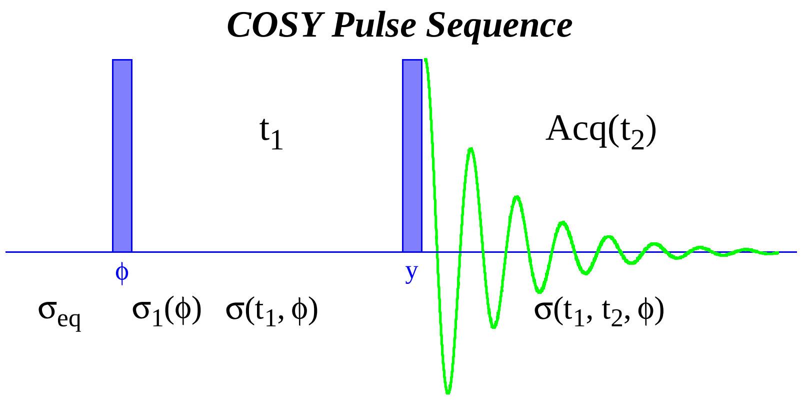


***Figure 0-12*** **A simple COSY pulse sequence. Density operators label key points in simulation of this exepriment.**

In this example, rather than incrementing the receiver phase (normal TPPI), for convenience the first pulse phase is incremented. No relaxation effects will be considered and the pulses taken as ideal. The resulting output file is produced in Felix format and subsequently processed with that program.

1. The TPPI detection scheme is discussed in Derome, *Modern NMR Techniques for Chemistry Research*, page 83.

## 1.6.2 Program

```
/*cosy.2.cc*******************************************************************-*-c++-*-
**                                                                                   **
**                    COSY Example Using Rectangular Pulses                          **
**                                                                                   **
** Simulates a cosy experiment using rectangular (soft) pulses. No phase             **
** cycling is employed. The data is processed using NMRi from Tripos.                **
**                                                                                   **
*************************************************************************************/

#include <gamma.h>
main()
{
const double dt1 = 0.001;                           // increments in τ1
const double dt2 = 0.001;                           // increments in τ2
const int pt1 = 1024;                               // acquisition points for τ1
const int pt2 = 1024;                               // acquisition points for τ2
const double puls_duration = 10.0e-6;
spin_system sys;                                    // define the system;
gen_op sigma, sigma1, H, detect;
block_1D tmp(pt2);                                  // variable for the response
block_2D data(pt1,pt2);                             // variable for the 2D-data
sys.read(”cosy.sys”);                               // read the system
sigma = sigma_eq(sys);                              // equilibrium density matrix
H = Hcs(sys) + HJ(sys);                             // Hamiltonian of the system)
detect = Fm(sys);
sigma = Sypuls(sys, sigma, H,“1H”,
                         0, puls_duration, 90);     // apply the rf puls,0 Hz offset
for (int step=0; step<pt1; step++)                  // loop for τ1
    {
    sigma1=evolve(sigma, H, step*dt1);              // evolution during t1
    sigma = Sypuls(sys, sigma, H,“1H”,
    0, puls_duration, 90);                          // the second pulse
    FID(sigma1, detect, H, dt2, pt2, tmp);          // record the FID
    data.put_block(step, 0, tmp);                   // store it in the 2D block
    }
NMRi (“cosy.dat”, data);                            // Output for the NMRi program
}
```

### 1.6.3 Results

The data was processed with the NMRi program. The following processing was done:

| | |
|---|---|
| EI | Set imaginary part to 0 |
| SB | Phased Sine Bell (0.5, 1, 6) |
| ZF | Zero fill (No) |
| FT | Fourier transformation |
| TP | Transpose |
| EI | Set imaginary part to 0 |
| SB | Phased Sine Bell (0.5, 1, 6) |

- ZFZero fill (No)
- FTFourier transformation

The result is shown at the end of the chapter (Spectrum 3.1.2)

# 2 E-COSY

## 2.1 Introduction

The E-COSY (Exclusive COrrelation SpectroscopY) experiment[1] is a COSY variation useful in the determination of scalar coupling constants which are not easily accessible through other more simplistic NMR experiments. As in COSY spectra, E-COSY spectra contain cross peaks between “active” spins - those that are scalar coupled. However, the cross peaks in E-COSY also exhibit additional shifts based on scalar couplings to other “passive” spins. In a couple network involving three spins, the structure of E-COSY cross peaks are easily rationalized from the structure of the COSY crosspeaks at the cross frequencies of any two of the spins involved. This is demonstrated in the following figure.

***E-COSY Cross Peaks for Three Coupled Spins***

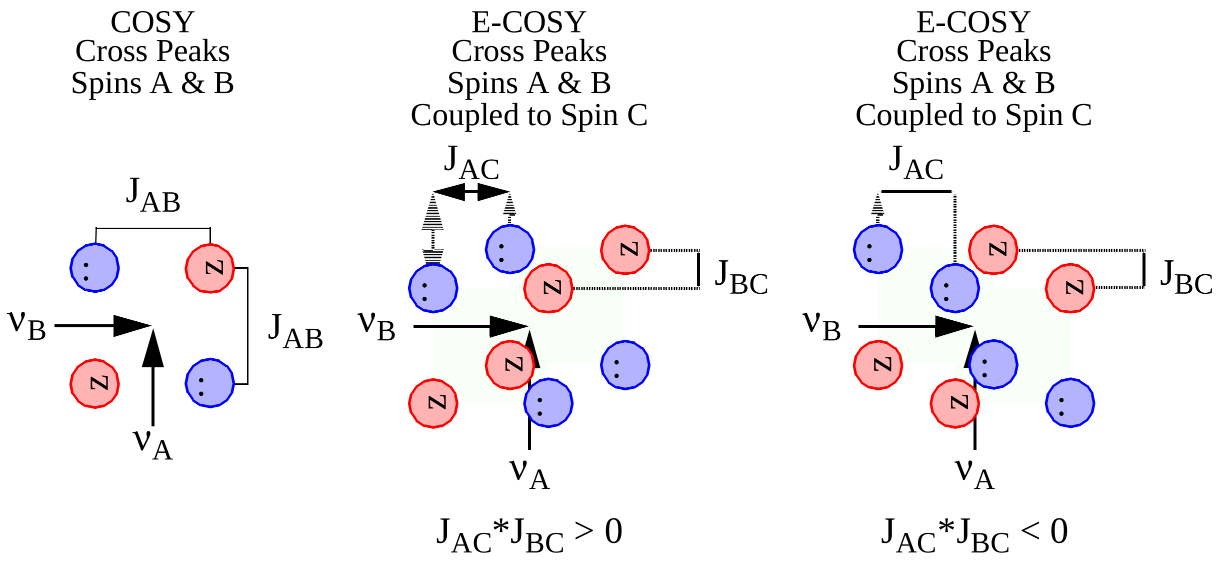


***Figure 0-13*** **The prototypical COSY cross peak between spins A and B which are scalar coupled exists of 4 peaks of alternating phase but pure absorption. In E-COSY spectra, if a third “passive” spin C is also mutually coupled to both A and B, the usual 4 COSY cross peaks are present twice, the two sets shifted by the two scalar coupling constants to C.**

As the spin coupling network increases in size the cross peak pattern becomes more complicated. Further problems will also result from overlaps, strong coupling, and relaxation effects. The following figure shows what is expected in a network of 4 mutually coupled spins.

1. C. Griesinger, O.W. Sørensen, and R.R. Ernst (1985), *JACS*, **107**, 6394-6396, “Two-Dimensional Correlation of Connected NMR Transitions”.
C. Griesinger, O.W. Sørensen, and R.R. Ernst (1986), *J. Chem. Phys.*, **85**, 6837-6851, “Correlation of connected transitions by two-dimensional NMR spectroscopy”.
C. Griesinger, O.W. Sørensen, and R.R. Ernst (1987), *JMR*, **75**, 474-492, “Practical Aspects of the E.COSY Technique. Measurement of Scalar Spin-Spin Coupling Constants in Peptides”.

## *E-COSY Cross Peaks for Four Coupled Spins*

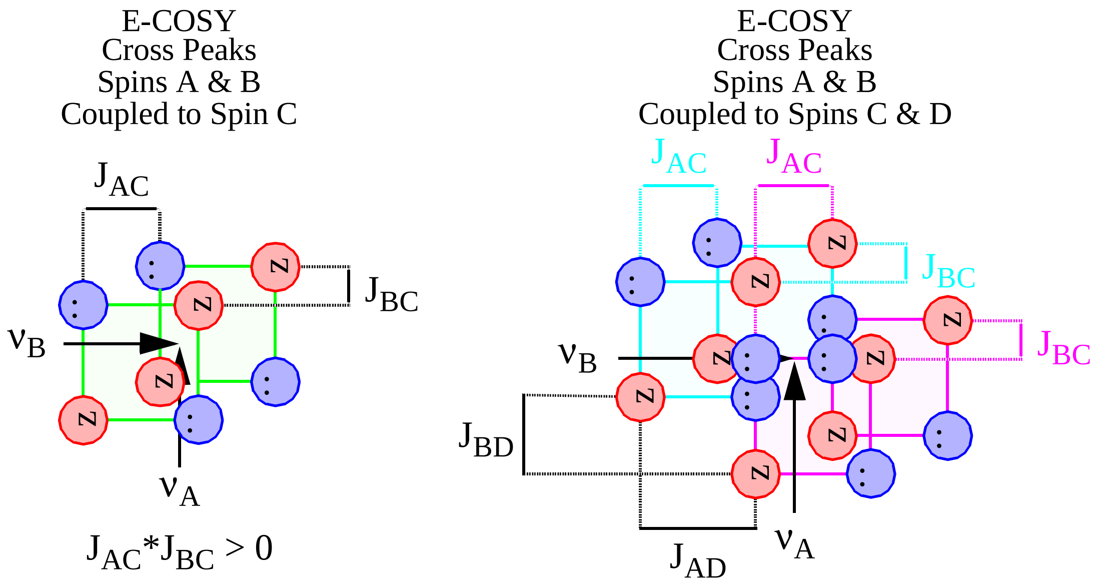
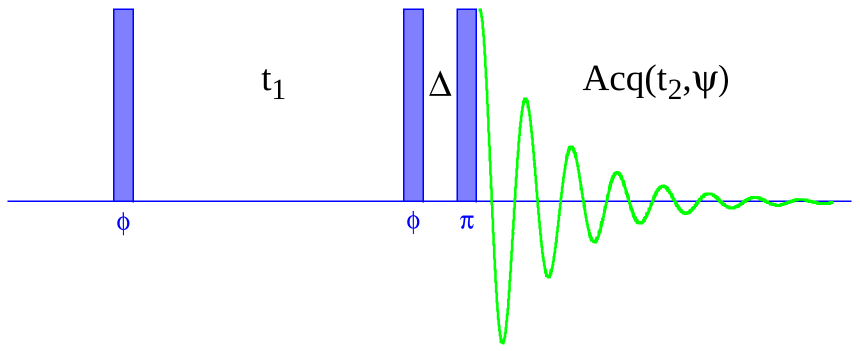


***Figure 0-14*** **The E-COSY cross peaks expected from four mutually coupled spins can be rationalized from the cross peaks in the three spin network. The eight peaks in the three spin system propagate into 2 sets of eight peaks which are shifted by the additional coupling constants to the two “active” spins A and B, namely $J_{BD}$ and $J_{AD}$.**

The pulse sequence and phase cycle for the E-COSY experiment are given in the following figure[1].

## *E-COSY Pulse Sequence and Phase Cycle*



| W | $\phi$ | $\psi$ |
|---|---|---|
| 4 | 0 | 0 |
| 1 | 120 | 0 |
| 1 | 240 | 0 |
| 3 | 60 | 180 |
| 3 | 300 | 180 |

***Figure 0-15*** **: Homonulcear E-COSY pulse sequence. The phase cycle is given in the previous JMR article in TABLE 1, page 477, with N=K=3.**

This E-COSY sequence is identical to that used for MQF-COSY experiments. In MQF-COSY the phase cycle is adjusted to select out the coherence order desired. The E-COSY experiment is simply a linear combination of MQF-COSY so there is an additionally “weighting factor” W listed in the phase cycle.

1. A alternative E-COSY sequence is given in the previous *J. Chem. Phys* reference, page 6839, FIG. 4. That sequence has only two pulses, both of constant phase with the second pulse having a cycled pulse angle.

## 2.2 E-COSY Examples

The following set of example programs are covered in this chapter. They simulate E-COSY experiments and/or E-COSY related experiments and values. Both homonuclear and heteronuclar spin systems are treated.

**Table 2: E-COSY Example Programs**

| Example | Page | Pulse[a] | Relaxation | System | Workup[b] |
|---|---|---|---|---|---|
| 1. E-COSY | 41 | Ideal | No | Homonuclear | Felix |
| 2. Alternate E-COSY | 46 | Ideal | No | Homonuclear | Felix |
| 3. Complimentary | 54 | Ideal | No | Homonuclear | Felix |
| 4. Heteronuclear E-COSY | 61 | Ideal | No | Heteronuclear | Felix |
| 5. E-COSY Phase Cycles | 69 | N/A | N/A | N/A | FrameMaker |
| 6. E-COSY with Relaxation | 76 | Ideal | Yes | Homonuclear | Felix |
| 7. E-COSY with Relaxation |  | Ideal | Yes | Heteronuclear | Felix |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

a. Application of an “Ideal” pulse is “infinitely” short so n relaxation effects can be considered in such steps.
b. Workups labeled “Open” allow the user to choose the output type while the program is running. The output types currently available in GAMMA are FrameMaker, Felix, NMRi, and MATLAB

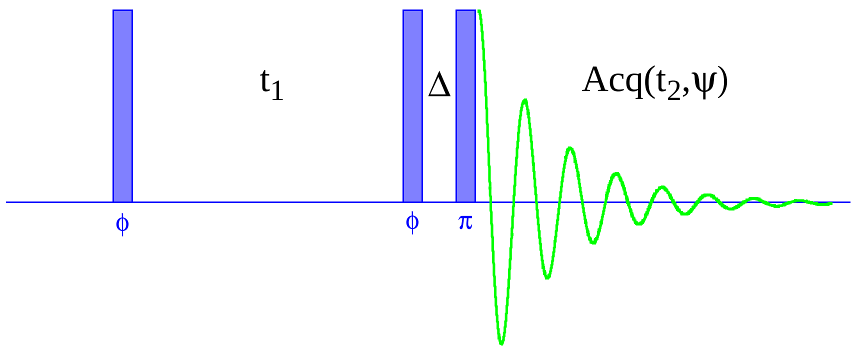
Each section in this chapter discusses at length a corresponding simulation in the above table. Although this document should contain listings of all files necessary to run the program, they should be included (reside on disk) with the GAMMA installation as well. These will lie in the */gamma/Example subdirectories.

## 2.3 E-COSY with Superoperators

### 2.3.1 Description

Our initial E- COSY simulation will implement the pulse sequence and phase cycle previously described. It is repeated below for reader clarity.

***E-COSY Pulse Sequence and Phase Cycle***

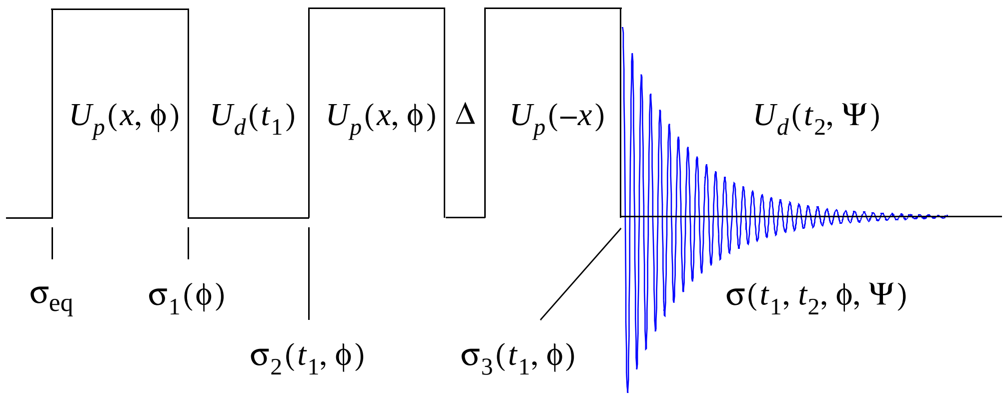


| W | $\phi$ | $\psi$ |
|---|---|---|
| 4 | 0 | 0 |
| 1 | 120 | 0 |
| 1 | 240 | 0 |
| 3 | 60 | 180 |
| 3 | 300 | 180 |

***Figure 0-16*** **The standard E-COSY pulse sequence with appropriate phase cycle and weighting factors. The phase angles are given by $\phi = j(\pi/N)$ with $j = [0, 2N-1]$ ; we have used N=3. The weighting factors are given by $W_0 = 4$ for j=0, $W_3 = 0$ for j=K=3, and $W_j = \frac{3}{4}\frac{(-1)^j}{\sin^2(\beta_j/2)}$ for all other j values. All formulae are obtained from the previous reference in JCP, page 6842. The weights are obtained with $B_0$=$B_1$=0 and the $W_j$ are scaled by a factor of two.**

In the figure, W is a weighting factor, $\phi$ the phase angle of the first two pulses, and $\psi$ the phase angle of the detector. The evolution of the density matrix for this E-COSY sequence is depicted below in terms of Hilbert space propagators.

***E-COSY Sequence in Terms of Hilbert Space Propagators***



Accordingly, the evolution of the density matrix during an E-COSY experiment is given in the following set of equations.

$$\sigma_1(\phi) = U_p(x,\phi)\sigma_{eq}U_p^{-1}(x,\phi) = R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_{eq}R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi) \quad \text{(EQ 1)}$$

$$\sigma_2(t_1,\phi) = U_d(t_1)\sigma_1(\phi)U_d^{-1}(t_1) \quad \text{(EQ 2)}$$

$$\begin{aligned}\sigma_3(t_1,\phi) &= U_p(-x)U_p(x,\phi)\sigma_2(t_1,\phi)U_p^{-1}(x,\phi)U_p^{-1}(-x)\\ &= U_p(-x)R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_2(t_1,\phi)R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x)\end{aligned} \quad \text{(EQ 3)}$$

$$\sigma(t_1,t_2,\phi) = U_d(t_2)\sigma_3(t_1,\phi)U_d^{-1}(t_2) \quad \text{(EQ 4)}$$

The FID is then computed by performing trace operations with a detection operator, $F_-$, which is also phase cycled.

$$FID(t_1,t_2,\phi,\psi) = Tr\{F_-(\psi)\sigma(t_1,t_2,\phi)\} \quad \text{(4-1)}$$

The last step is that we sum the FID's over the phase cycle

$$FID(t_1,t_2) = \sum_{\phi,\psi}^{\text{cycle}} FID(t_1,t_2,\phi,\psi) \quad \text{(4-2)}$$

With GAMMA, we could proceed implementing these equations as outlined and directly perform an E-COSY simulation. Such a task would require some scheme for performing the phase cycle, perhaps involving a complicated algorithm and definitely involving repetitive computations. Thus, rather than using these six equations as written, we will shuffle around the formulae and attempt to isolate components involving the phase cycle from components involving the $t_1$ and $t_2$ time increments. The result will simplify implementation of the E-COSY sequence. From the first two equations we have

$$\sigma_2(t_1,\phi) = U_d(t_1)R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_{eq}R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi)U_d^{-1}(t_1) \quad .$$

Rotations about the z-axis cannot affect the equilibrium density matrix, so the inner rotation operators may be safely removed. Furthermore, rotations about z commute with the time evolution propagators due to $[H_0,F_z] = 0$, so we may switch the order of these operations. The result is

$$\sigma_2(t_1,\phi) = R_z(\phi)U_d(t_1)U_p(x)\sigma_{eq}U_p^{-1}(x)U_d^{-1}(t_1)R_z^{-1}(\phi) \quad . \quad \text{(4-3)}$$

We define a new density matrix operators which are independent of the phase angle $\phi$,

$$\sigma_1 = U_p(x)\sigma_{eq}U_p^{-1}(x) \quad \text{(4-4)}$$

and

$$\sigma_2(t_1) = U_d(t_1)\sigma_1U_d^{-1}(t_1) \quad , \quad \text{(4-5)}$$

so that the previous formula for $\sigma_2(t_1, \phi)$ , (4-3), becomes simply

$$\sigma_2(t_1, \phi) = R_z(\phi)\sigma_2(t_1)R_z^{-1}(\phi) \quad . \tag{4-6}$$

Keep in mind our objectives: to separate time evolution steps from phase cycle steps. Clearly, equation (4-6) shows how to obtain $\sigma_2(t_1, \phi)$ from independent phase and time evolutions steps.

A cancellation of the rotation operators in (4-6) occurs when the next step is formulated. From (EQ 3),

$$\sigma_3(t_1, \phi) = U_p(-x)R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_2(t_1, \phi)R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x)$$

$$\sigma_3(t_1, \phi) = U_p(-x)R_z(\phi)U_p(x)\sigma_2(t_1)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x) \tag{4-7}$$

Again, note that the phase cycle is explicitly removed from the t1 incrementation at this point and we are at the last step in the pulse sequence, detection.

Detection involves propagation for the $t_2$ delay followed by multiplication with a detection operator respectively. We have

$$F_-(\psi)\sigma(t_1, t_2, \phi) = R_z(\psi)F_-R_z^{-1}(\psi)\sigma(t_1, t_2, \phi) \quad . \tag{4-8}$$

and can utilize the relationship $Tr\{AB\} = Tr\{BA\}$ , to produce

$$\begin{aligned} Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} &= Tr\{R_z(\psi)F_-R_z^{-1}(\psi)\sigma(t_1, t_2, \phi)\} \\ &= Tr\{F_-R_z^{-1}(\psi)\sigma(t_1, t_2, \phi)R_z(\psi)\} \end{aligned}$$

If the delay $t_2$ is explicitly written we obtain

$$\begin{aligned} Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} &= Tr\{F_-R_z^{-1}(\psi)U_d(t_2)\sigma_3(t_1, \phi)U_d^{-1}(t_2)R_z(\psi)\} \\ &= Tr\{F_-U_d(t_2)R_z^{-1}(\psi)\sigma_3(t_1, \phi)R_z(\psi)U_d^{-1}(t_2)\} \end{aligned} \tag{4-9}$$

Thus, the phase cycle over $\psi$ has been isolated from the detection and $t_2$ incrementation. If we again focus on the components which depend upon the phase cycle, we have

$$\begin{aligned} R_z^{-1}(\psi)\sigma_3(t_1, \phi)R_z(\psi) &= \sigma_3(t_1, \phi, \psi) \\ &= R_z^{-1}(\psi)U_p(-x)R_z(\phi)U_p(x)\sigma_2(t_1)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x)R_z(\psi) \end{aligned} \quad . \tag{4-10}$$

The entire E-COSY pulse sequence phase cycle is contained in the above equation and we have fulfilled our objective of completely separating the time incrementation from the phase cycle mathematically. We can formulate a new propagator which contains the phase cycle. Letting

$$U_{mix}(\phi, \psi) = R_z^{-1}(\psi)U_p(-x)R_z(\phi)U_p(x) \quad , \tag{4-11}$$

the trace equation becomes,

$$\begin{aligned} Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} &= Tr\{F_-U_d(t_2)U_{mix}(\phi, \psi)\sigma_2(t_1)U_{mix}^{-1}(\phi, \psi)U_d^{-1}(t_2)\} \\ &= Tr\{F_-U_d(t_2)\sigma_3(t_1, \phi, \psi)U_d^{-1}(t_2)\} \end{aligned} \quad . \tag{4-12}$$

To clarify what has been accomplished, the E-COSY pulse sequence (for mathematical implementation, not experimental implementation) is rewritten in the following diagram to show the distinction between the time delays and the phase cycle steps.

***Equivalent E-COSY Sequence in Terms of Hilbert Space Propagators***

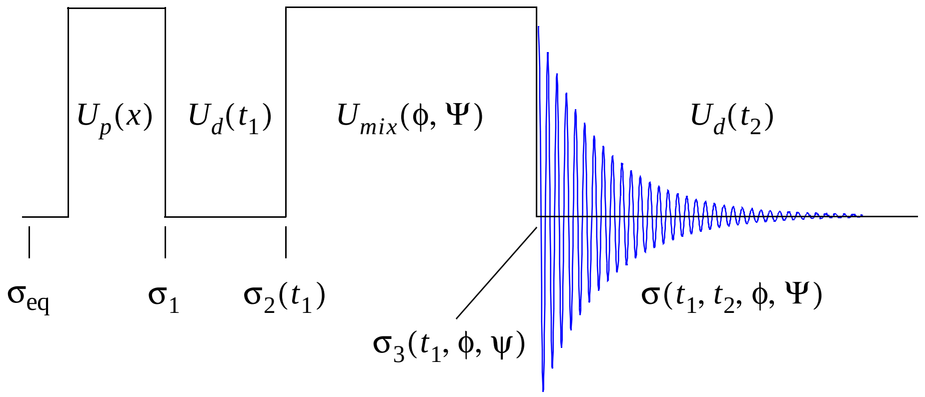


***Figure 0-17*** **The E-COSY pulse sequence in terms of adjusted Hilbert space propagators. At this point the propagators for the delays are separate from the propagators involving the phase angles.**

Of course, in order to compute our 2-dimensional E-COSY spectrum we need to sum FID's over each phase cycle according to (4-2)

$$FID(t_1, t_2) = \sum_{\phi, \psi}^{\text{cycle}} FID(t_1, t_2, \phi, \psi)$$

and we have a different mixing propagator for each combination of $\phi$ and $\psi$.

We now employ the power of superoperators and it will hopefully become clear to the reader why we have bothered rewriting the mathematical steps describing E-COSY. We shall form the unitary transformation superoperator equivalent to the mixing propagator. The equation then becomes,

$$Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} = Tr\left\{F_-U_d(t_2)[\hat{\Gamma}_{mix}(\phi, \psi)\sigma_2(t_1)]U_d^{-1}(t_2)\right\} \quad . \tag{4-13}$$

and we can take advantage of superoperator linearity in steps which are not linear in a Hilbert space propagator formulation. Now when we sum over the phase cycle we have

$$FID(t_1, t_2) = \sum_{\phi, \psi} FID(t_1, t_2, \phi, \psi)$$

$$FID(t_1, t_2) = \sum_{\phi, \psi}^{cycles} Tr\left\{F_- U_d(t_2)[\hat{\Gamma}(\phi, \psi)\sigma_2(t_1)]U_d^{-1}(t_2)\right\}$$

$$FID(t_1, t_2) = Tr\left\{F_- U_d(t_2)\left[\sum_{\phi, \psi}^{cycles} \hat{\Gamma}(\phi, \psi)\sigma_2(t_1)\right]U_d^{-1}(t_2)\right\} \tag{4-14}$$

$$FID(t_1, t_2) = \left\{F_- U_d(t_2)[\hat{\Gamma}_{cycle}(\phi, \psi)\sigma_2(t_1)]U_d^{-1}(t_2)\right\}$$

The phase cycle needed for the E-COSY simulation is now entirely contained in the superoperator, and may be applied in a single step. On a computational level, this means that the phase cycling loop can be removed from the loop over $t_1$ increments. The code becomes more concise and the computation more efficient[1]. The E-COSY pulse sequence in terms of the phase cycle superoperator can depicted as follows.

**_E-COSY Sequence Using A Superoperator Phase Cycle_**

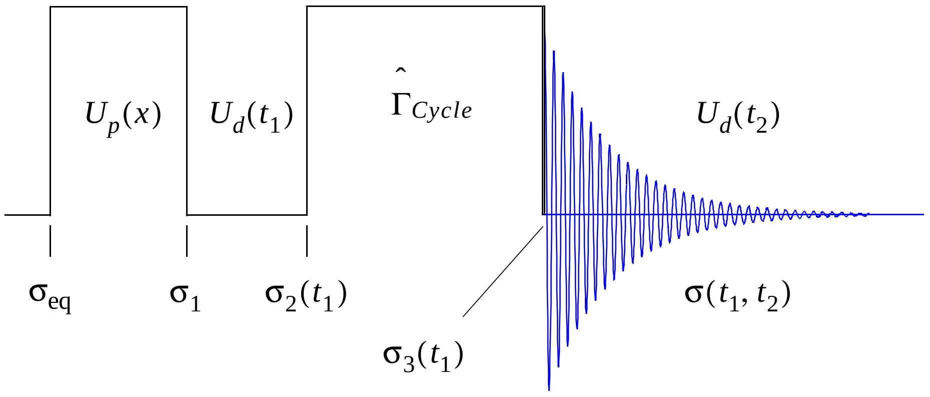


***Figure 0-18*** **The E-COSY pulse sequence in terms of Hilbert space propagators and a superoperator for the phase cycle. The entire phase cycle which is essential for E-COSY has been replaced by a single evolution under a Liouville space superoperator.**

The simulation will be performed exactly this in this manner. We initially form the equilibrium density matrix, the pulse propagators, the $t_1$ propagator for $t_1$ incrementation time, and the superoperator representing the entire E-COSY phase cycle.

1. This is true only for small spin systems, <5 spins. As the system size increases it becomes more difficult for computers to handle superoperators due to the size of the Liouville space.

## *Program Outline for the E-COSY Sequence*

| **Step** | | **Density Matrix** |
|---|---|---|
| **1.** | ***Form Equilibrium Density Matrix*** | $\sigma_{eq}$ |
| **2.** | ***Apply the 1st x-pulse*** | $\sigma_1$ |
| **3.** | ***Evolve for $t_1$ time*** | $\sigma_2(t_1)$ |
| **4.** | ***Evolve under E-COSY "Phase Cycle"*** | $\sigma_3(t_1)$ |
| **5.** | ***Acquire FID for this $t_1$ time*** | $\sigma(t_1, t_2)$ |
| **6.** | ***Repeat Steps 3-6 for each $t_1$ time*** | |

## 2.3.2 Program

```
/* ecosy1.cc *******************************************************************-*-c++-*-
**                                                                              **
**                    Example Program for the GAMMA Library                     **
**                                                                              **
** This program simulates an E-COSY experiment with complete phase              **
** cycling. The pulses are ideal and relaxation effects are not considered.     **
**                                                                              **
** See: C. Griesinger, O.W. Sorensen, and R.R. Ernst, JMR, 75, 474-492,         **
** (1987), “Practical Aspects of the E.COSY Technique. Measurement of           **
** Scalar Spin-Spin Coupling Constants in Peptides”. The required phase         **
** cycle is found on page 477 of this reference.                                **
**                                                                              **
** Note: This program treats only homonuclear spin systems.                     **
** Superoperators are used to implement the phase cycle                         **
** which speeds up the computation for small systems                            **
** but this program is cumbersome for large systems.                            **
**                                                                              **
******************************************************************************** */

#include “gamma.h”
//                         Define Constants

 const P_cycl = 12;                             // Phase cycle length
 const double P_mix[12] =                       // Pulse phase cycle
        {0,60,0,60,0,60,0,300,120,300,240,300};
//                         Begin Program

main (int argc, char* argv[])
{
 cout << “\n\tSimulation of an E-COSY spectrum in TPPI mode\n”;
 cout << “\t\t (Hard Pulses, No Relaxation)\n”;
                           Read in Parameters

String filename;                                // Name of spin system file
 query_parameter(argc, argv, 1,                 // Get filename from command
        “\n\tSpin system filename? “, filename); // line or ask for it
 spin_system sys;                               // Declare spin system sys
 sys.read(filename);                            // Read system from filename
 int t1pts, t2pts;
 query_parameter(argc, argv, 2,                 // Get number FID of points
             “\n\tAcquisition Size? “, t2pts);
 t1pts = 2*t2pts;                               // Set t1 size for TPPI
 String J;
 query_parameter(argc, argv, 3,                 // Weak or strong coupling
        “\n\tWeak or strong coupling (w/s)? “, J);
 double offset = query_offset(sys, 0, 1);       // Ask for an offset frequency
 double NyqF = query_Nyquist(sys, 0);           // Choose a Nyquist frequency
 double t2dt = 1.0/(2.0*NyqF);                  // Dwell time, quadrature
 double t1dt = t2dt/2.0;                        // t1 time increment, TPPI

                     Set Up Operators, Superoperator
 gen_op H;                                      // Set Hamiltonian for
 if (J == “w”) H = Hcs(sys) + HJw(sys)          // Strong or weak coupling
else       H = Hcs(sys) + HJ(sys);
 gen_op Ud1 = Rz(sys,+90)*prop(H,t1dt);         // Delay t1 + TPPI propagator
 gen_op D = Fm(sys);                            // Detector to F-
 gen_op Upx = Ixypuls_U(sys,0.,90.);            // Propagator for x pulse
 gen_op Upmx = Ixypuls_U(sys,0.,-90.);          // Propagator for -x pulse
 gen_op U_mix;                                  // Temporary mixing propagator
 super_op G_mix;                                // Phase cycle superoperator
 gen_op sigma3;
//              Construct Mixing, Phase Cycle Superoperator

 double conv = acos(-1)/180;
 double P_det = 0;
 cout << “\n\tE-COSY Phase Cycle\n”;
 cout << “\n\t# scan                              betareference\n”;
 for ( int i=0; i<P_cycl; i++ )
  {
  cout << “\t” << i << “\t”<< P_mix[i]          // Output E-COSY phase
        << “\t” << P_det << “\n”;
  U_mix = Rz(sys,-P_det);                       // Detector phase cycle
  U_mix *= Upmx;                                // 3rd 90 Pulse (-x)
  U_mix *= Rz(sys,+P_mix[i]);                   // Phase shift pulses 1 & 2
  U_mix *= Upx;                                 // 2nd 90 Pulse
  U_mix.Op_base(H);                             // Put in eigenbasis of Ho
  G_mix += U_transform(U_mix);                  // Add to U transform superop
  P_det = acos(-cos(P_det*conv))/conv;          // Adjust detector phase
  }
//                        Apply Pulse Sequence

 File ecosy;                                    // Declare and open file
 ecosy.open(“ecosy.dat”,
       io_writeonly, a_create);
 block_1D t2BLK(t2pts);                         // Set 1D block for output
 gen_op sigma0 = sigma_eq(sys);                 // Set density matrix equilibrium
 gen_op sigma1 = evolve(sigma0,Upx);            // Apply first (PI/2)x pulse
 gen_op sigma2 = sigma1;                        // Initial sigma2 (t1 = 0)
  for (int t1=0; t1<t1pts; t1++)                // Loop over all t1 increments
  {
  sigma3 = evolve(sigma2, G_mix);               //superop phase cycle
  FID(sigma3,D,H,t2dt,t2pts,t2BLK);             //acquisition
  Felix(ecosy, t2BLK);                          //output block: Felix
  evolve_ip(sigma2, Ud1);                       //evolution next t1
  }
 ecosy.close();                                 // Close file
 double Om = sys.Omega();                       // Spectrometer frequency
 Felix2D_params(cout, Om, 2.0*NyqF,
            2*t2pts, offset);                   // Output Felix parameters
 cout << “\n\n”;                                // Keep screen nice
}
```

### 2.3.3 Discussion

**Define Constants:** In this program the E-COSY phase cycle are explicitly defined; A 12 step cycle and associated phases in accordance with Figure 0-16.

**Read in Parameters:** As in prior simulations the spin system is read in from a disk file, the name may be supplied directly when running the program. The user is also asked for the acquisition size, whether to use a strong or weak coupling Hamiltonian, any desired offset, and a Nyquist frequency. Note that because TPPI will be used for phase sensitivity along the t1/f1 axis, twice as many points are taken along this axis that along the t2 axis, and the t1 incrementation time is half of the dwell time.

**Set up Operators, Superoperator:** The Hamiltonian is first formulated for either strong or weak coupling depending upon the input choice. Next, the $t_1$ time increment propagator is formulated. Here, in order to produce an acquisition with TPPI, a 90 degree rotation about the z-axis is added so that each incremented step includes the TPPI phase shift. The detector is set to F- in the next line. Propagators for the two 90 pulses are now computed, the first along x and the second along -x. A temporary operator is declared for used in summing over the phase cycle. The next line declares a superoperator which will ultimately account for the phase cycle. Finally, a working density matrix is specified.

**Construct Mixing, Phase Cycle Superoperator:** As outlined mathematically at the start of this section, the entire phase cycle can be removed from the t1 and t2 looping by use of a superoperator. In this section, the phase cycle loop is applied and the appropriate superoperator constructed. The loop goes over the twelve steps of the cycle (defined in the constants section) and at each step the phases are output to the terminal. Initially the detector phase is set to zero. The propagators for each step are then multiplied together <u>in the opposite order of the pulse sequence</u> due to the ordering of the unary *= step. At the end of each cycle, the unitary transform superoperator is summed to account for the mixing propagator. The last step in this section adjusts the detector phase to +/- 180.

**Apply Pulse Sequence:** Since the superoperator intrinsically contains the phase cycling, the pulse sequence application is quite simple. A file name ecosy is created and opened. A 1D data block called t2BLK is constructed to temporarily store the FID. Following the pulse sequence diagram, the initial density matrix is set to equilibrium. The next step is to apply a 90 degree pulse (without phase cycling) along the x-axis. This is evolved next by the propagator which accounts for $t_1$ incrementation as well as TPPI phase adjustments, but it is not done on the first step through the $t_1$ increments, at $t_1$=0. Rather, this step is done at the end of the loop. The next step is evolution under the superoperator which contains all the phase cycling. This density matrix, sigma3, is then used to simulate an FID which is subsequently sent to the file ecosy.dat in Felix format. Again, the last step of the loop is actually the earlier t1 incrementation. The last program steps close the file, get the spectrometer frequency, output Felix parameters necessary for spectral workup, and clear the screen.

## 2.3.4 Spin System

For this example we will first treat a simple 3 spin proton system in order to reproduce the published simulations[1] and verify that our program is producing the proper results.

***Initial 3-Spin Proton System for E-COSY Verification***

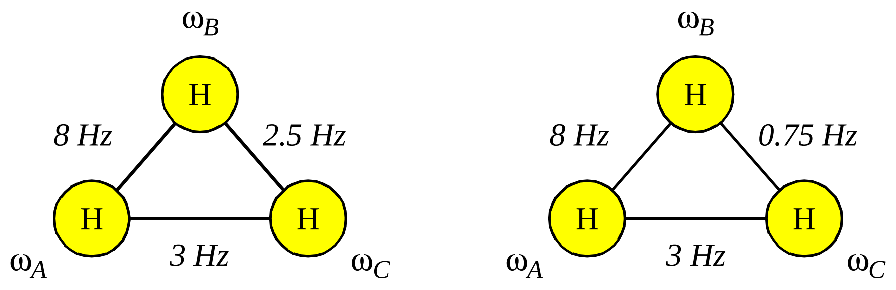


***Figure 0-19*** **Simple 3 spin protons systems to be utilized in verification of the E-COSY simulation program.**

**ecosy1A.sys**

```
SysName  (2) : ECOSY1A    - Name of the Spin System
NSpins   (0) : 3          - Number of Spins in the System
Iso(0)   (2) : 1H         - Spin Isotope Type
Iso(1)   (2) : 1H         - Spin Isotope Type
Iso(2)   (2) : 1H         - Spin Isotope Type
PPM(0)   (1) : -.1        - Chemical Shifts in PPM
PPM(1)   (1) : 0.0        - Chemical Shifts in PPM
PPM(2)   (1) :  .1        - Chemical Shifts in PPM
J(0,1)   (1) : 8.0        - Coupling Constants in Hz
J(0,2)   (1) : 3.0        - Coupling Constants in Hz
J(1,2)   (1) : 2.5        - Coupling Constants in Hz
Omega    (1) : 500        - 1H Spectrometer Frequency in MHz
```

ecosy1B.sys = ecosy1A.sys with replacements below

```
J(1,2)   (1) : -.75       - Coupling Constants in Hz
```

Afterwards we shall examine a system of both 3 and 4 spins which are meant to represent a prototypical amino acid.

## 2.3.5 Workup

The 2D-data sets produced from this simulation were processed with the program Felix. Each data set was subjected to the following Felix macro, called ecosy1.mac (the !comments to the right are not to be included in the macro - these are no longer allowed in the current Felix version.). Note that this macro ***assumes that the simulation had an acquisition size of 512 points.***

1. C. Griesinger, O.W. Sørensen, and R.R. Ernst (1987), *JMR*, **75**, 474-492, “Practical Aspects of the E.COSY Technique. Measurement of Scalar Spin-Spin Coupling Constants in Peptides”. These systems are those used in this paper to generate their E-COSY spectra in FIG. 15., page 488.

```
def datfile ecosy           ! simulated data file name “ecosy.dat”
def matfile ecosy           ! matrix file name “ecosy.mat”
def t1max 1024              ! dimension t1 size
def t2max 512               ! dimension t2 size
def total 1024
ty Opening source file “ &datfile “ .....! comment showing
which data file
ty Opening matrix file “ &matfile “ .....! comment showing
which matrix file
cmx                         ! close any existing matrix
cl                          ! close any existing dat files
mat &matfile write          ! open matrix ecosy.mat
ty Filling matrix with data ...! comment that data processing
starting
lb 4                        ! set line broadening for apodization
ph0 90                      ! set zero order phase correction
ph1 0                       ! set first order phase correction
ty Complex-FT of t2 into F2 ...! comment starting F2 transfor-
mations
for row 1 &t1max            ! loop through all t1 rows
re &datfile                 ! read block of ecosy.dat file
si &t2max                   ! set the size
em                          ! exponential multiplication
sb &t2max 90                ! sine bell windowing function
zf &total                   ! zero fill to total size
ft                          ! complex FFT
red                         ! reduce data to real
sto 0 &row                  ! store processed data into matrix
next                        ! loop back for next block
ty Real-FT of t1 into F1    ! comment for t1 workup
for col 1 &total            ! loop through all columns
loa &col 0                  ! retrieve column from matrix
si &t1max                   ! specify the size
em                          ! exponential multiplication
sb &t1max 90                ! sine bell
zf 2048                     ! zero fill (will loose with real fft)
rft                         ! real FFT
ph                          ! apply phase adjustment
red                         ! reduce data to real
sto &col 0                  ! store back into matrix
next                        ! get next column
ty Plotting contours .....  ! comments for processing point
ty zooming cross peaks ..
cmx                         ! clear matrices
mat &matfile read           ! open ecosy.mat for reading only
lim 1 1 1024                ! set plot limits
lim 2 1 1024
lvl 5e-4                    ! set scaling
cpn 1                       ! contour positive and negative peaks
pen 1                       ! set first pen to pen 1
nl 4                        ! four contour levels
cyc 2                       ! pen cycle
cp                          ! contour plot
ty Processing finished ..   ! comment for processing end
end
```

Before running this macro, construct matrix ecosy.mat with cmd

```
bld ecosy 2 1024 1024 0! real array
```

which constructs a 2 dimensional array (1K x 1K) of real numbers. After processing the matrix axes are set with the rmx command (used twice, once for each axis).

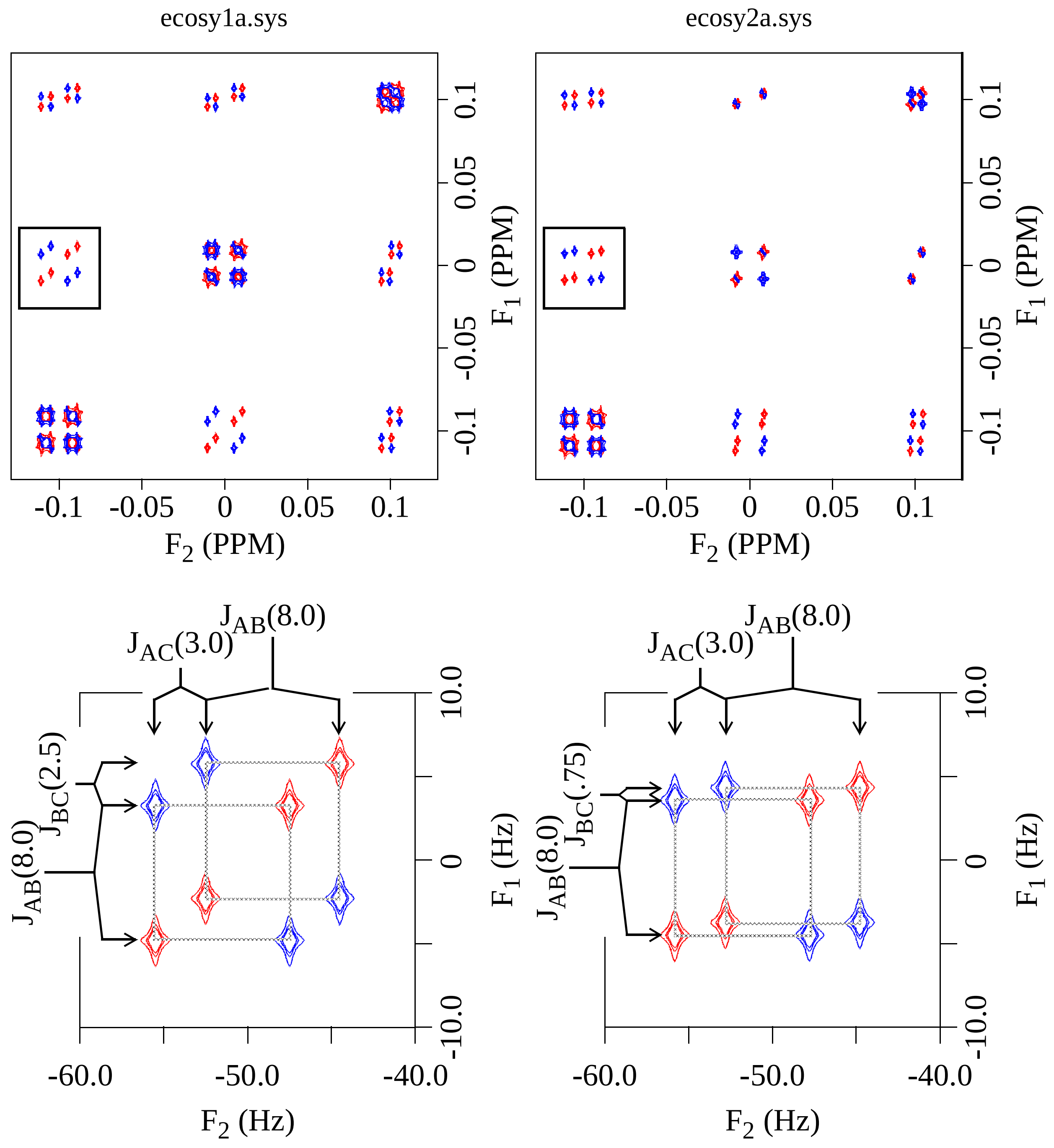
```
rmx
     1(2)                   ! specify the dimension (1 or 2)
     500                    ! spectrometer frequency
     128                    ! spectral width
     3                      ! PPM plotted on axis
     512                    ! reference point
     0                      ! reference point value
     F2(F1)                 ! axis label (F2 or F1)
```

The simulated E-COSY spectra were placed into this FrameMaker document in encapsulated Postscript format which resulted from sending the HPGL output of Felix through the filter program hpgl-toeps provided by FrameMaker. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
hdv felix.hpgl              ! hardcopy device is
file felix.hpgl
hpm 32                      ! hardcopy plot mode
is hpgl
hcp                         ! produce the plot
```

## 2.3.6 Results

***Simulated E-COSY Spectra on a 3-Spin Proton System***



The simulation was repeated for the two 3-spin systems, the input spin systems files are used to label the plots. Each plot was output from Felix, placed into FrameMaker MIF format, rescaled and annotated, and finally placed onto this page. All were performed with weak coupling.

### 2.3.7 Spin System

The E-COSY simulation will now be run on a number of 3- and 4- spin system which contain protons and coupling constants for the prototypical amino acid system in the next figure.

***3,4-Spin "Amino Acid" Proton System for E-COSY Simulations***

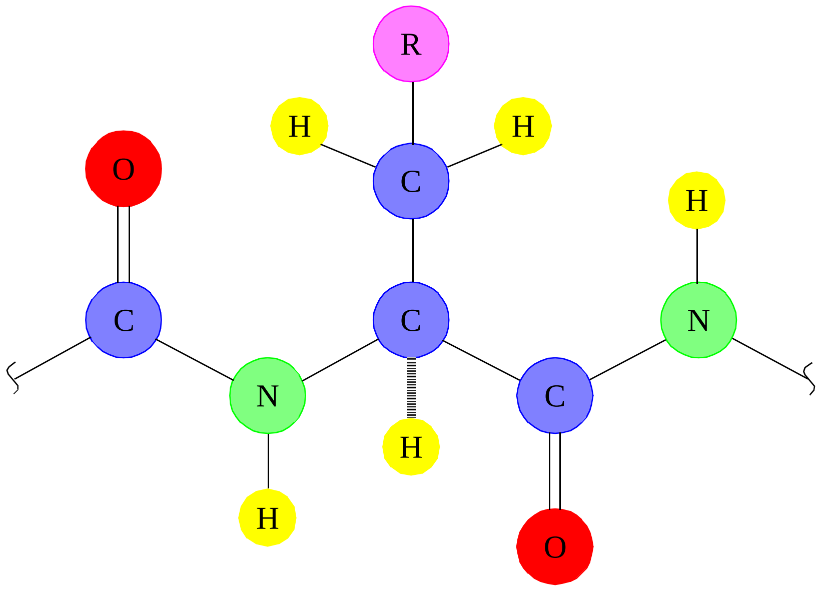


*Figure 0-20* **The spin system network which will be used in the E-COSY simulation examples.**

Only two of the systems are given in full, the others are derived from these.

**ecosy13A.sys**

| | | |
|---|---|---|
| **SysName** | **(2) : AMX** | **- Name of the Spin System** |
| **NSpins** | **(0) : 3** | **- Number of Spins in the System** |
| **Iso(0)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(1)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(2)** | **(2) : 1H** | **- Spin Isotope Type** |
| **PPM(0)** | **(1) : -.420** | **- Chemical Shifts in PPM** |
| **PPM(1)** | **(1) : -.035** | **- Chemical Shifts in PPM** |
| **PPM(2)** | **(1) : .385** | **- Chemical Shifts in PPM** |
| **J(0,1)** | **(1) : 6.8** | **- Coupling Constants in Hz** |
| **J(0,2)** | **(1) : 0.0** | **- Coupling Constants in Hz** |
| **J(1,2)** | **(1) : 9.8** | **- Coupling Constants in Hz** |
| **Omega** | **(1) : 256** | **- Spectrometer Frequency in MHz (1H based)** |

**ecosy13B.sys = ecosy13A.sys with replacements below**

| | | |
|---|---|---|
| **J(0,2)** | **(1) : -.9** | **- Coupling Constants in Hz** |

**ecosy13C.sys = ecosy13A.sys with replacements below**

| | | |
|---|---|---|
| **J(0,2)** | **(1) : -1.9** | **- Coupling Constants in Hz** |

**ecosy13D.sys = ecosy13A.sys with replacements below**

| | | |
|---|---|---|
| **J(0,2)** | **(1) : 4.9** | **- Coupling Constants in Hz** |

**ecosy14A.sys**

| | | |
|---|---|---|
| **SysName** | **(2) : system** | **- Name of the Spin System** |
| **NSpins** | **(0) : 4** | **- Number of Spins in the System** |
| **Iso(0)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(1)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(2)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(3)** | **(2) : 1H** | **- Spin Isotope Type** |
| **PPM(0)** | **(1) : -.420** | **- Chemical Shifts in PPM** |
| **PPM(1)** | **(1) : -.035** | **- Chemical Shifts in PPM** |
| **PPM(2)** | **(1) : .250** | **- Chemical Shifts in PPM** |
| **PPM(3)** | **(1) : .385** | **- Chemical Shifts in PPM** |
| **J(0,1)** | **(1) : 6.8** | **- Coupling Constants in Hz** |
| **J(0,2)** | **(1) : 0.0** | **- Coupling Constants in Hz** |
| **J(0,3)** | **(1) : 0.0** | **- Coupling Constants in Hz** |
| **J(1,2)** | **(1) : 9.8** | **- Coupling Constants in Hz** |
| **J(1,3)** | **(1) : 0.0** | **- Coupling Constants in Hz** |
| **J(2,3)** | **(1) : -15** | **- Coupling Constants in Hz** |
| **Omega** | **(1) : 256** | **- Spectrometer Frequency in MHz (1H based)** |

**ecosy14B.sys = ecosy14A.sys with replacements below**

| | | |
|---|---|---|
| **J(0,3)** | **(1) : -.9** | **- Coupling Constants in Hz** |

**ecosy14C.sys = ecosy14A.sys with replacements below**

| | | |
|---|---|---|
| **J(1,3)** | **(1) : 3.9** | **- Coupling Constants in Hz** |

**ecosy14D.sys = ecosy14A.sys with replacements below**

| | | |
|---|---|---|
| **J(0,3)** | **(1) : -.9** | **- Coupling Constants in Hz** |
| **J(1,3)** | **(1) : 3.9** | **- Coupling Constants in Hz** |

### 2.3.8 Workup

The 2D-data sets were processed in the same manner as for the previous plots except that the data matrices were kept smaller. Due to the inherent limitations in the Felix macro used, a second macro is supplied, ecosy1b.mac which ***assumes that the simulation had an acquisition size of128 points.***

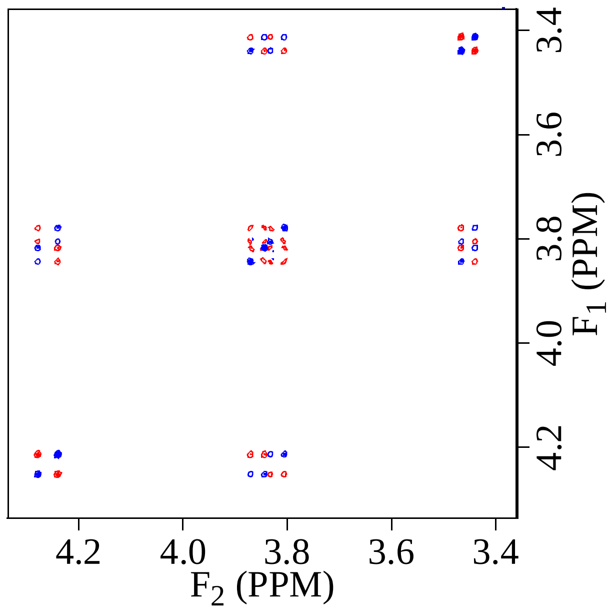
Prior to running this Felix macro the matrix ecosy.mat must be constructed with the following Felix command

```
bld ecosy 2 256 256 0                    ! real array
```

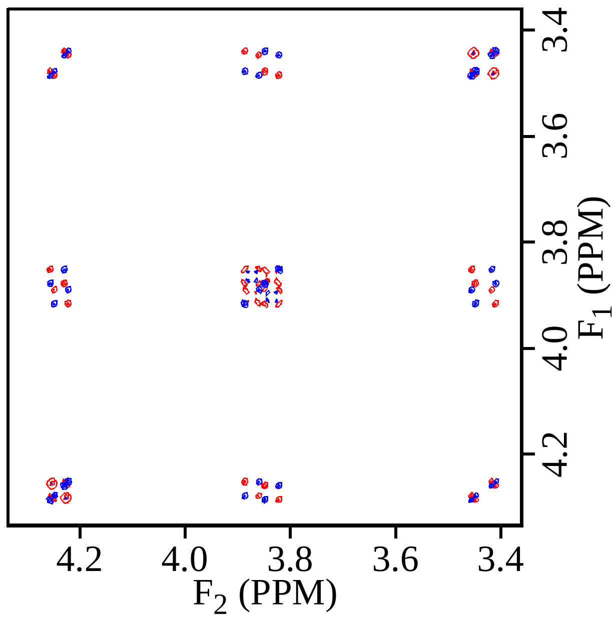
## 2.3.9 Results

***Simulated E-COSY Spectra on a 3,4-Spin "Amino Acid" Proton System***

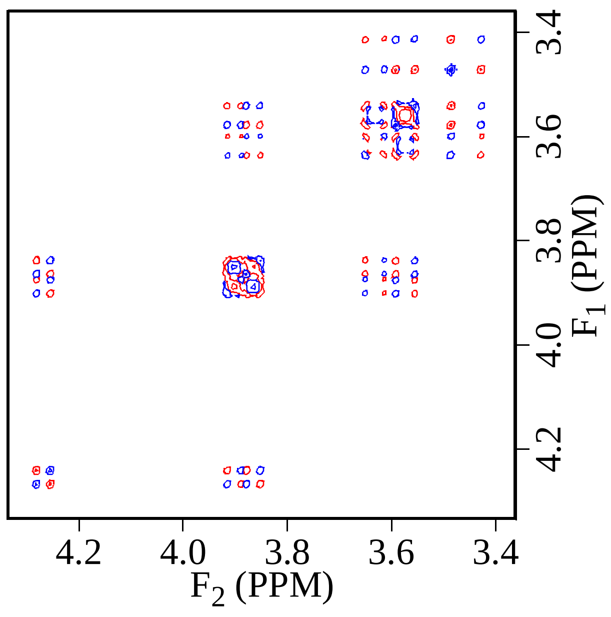
ecosy13a.sys



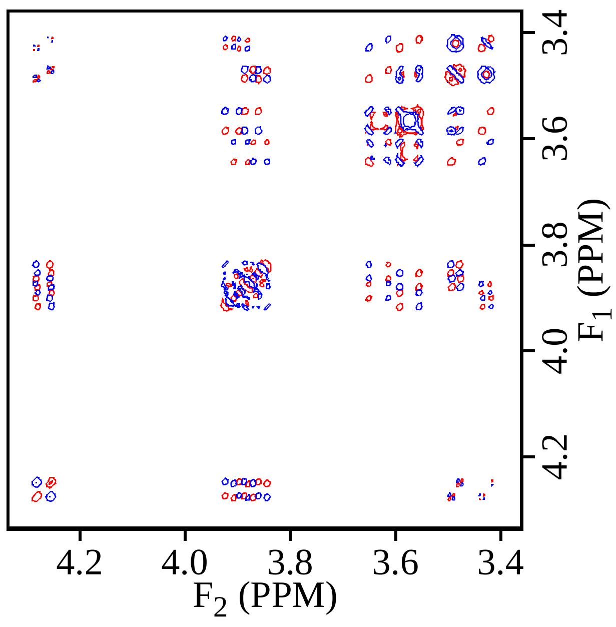
ecosy13b.sys



ecosy14a.sys



ecosy14b.sys

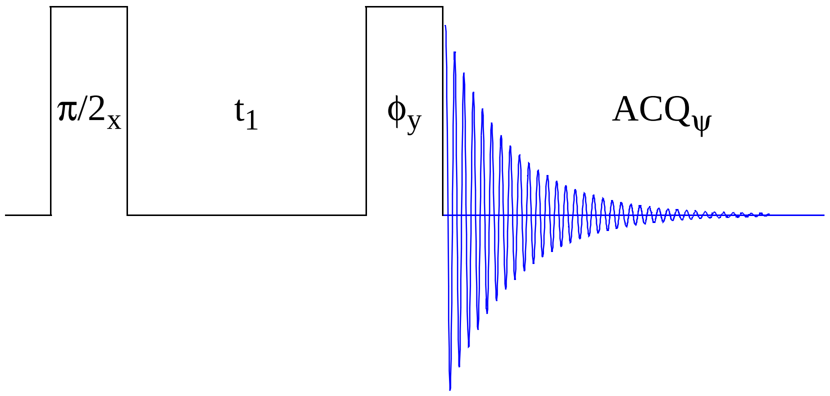


## 2.4 Alternative E-COSY

### 2.4.1 Description

An alternative pulse sequence has been given in the literature as being equivalent to the E-COSY sequence used in the last simulation[1]. The pulse sequence is more simplistic and, although perhaps more difficult to implement experimentally, readily implemented in a GAMMA program.

***Alternative E-COSY Pulse Sequence***

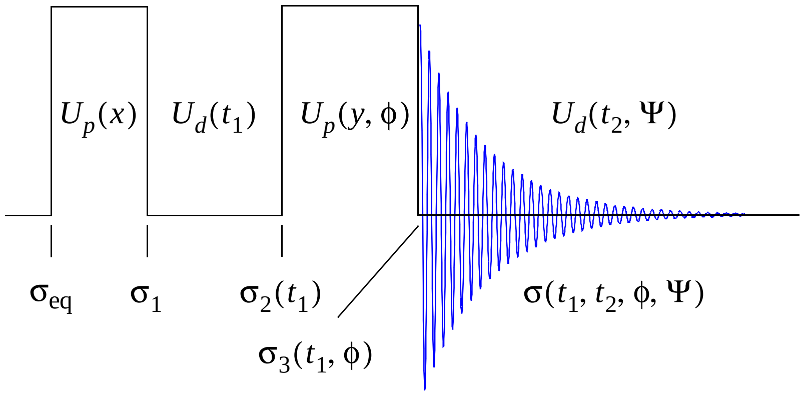


| W | $\phi$ | $\psi$ |
|---|---|---|
| 4 | 0 | 0 |
| 1 | 120 | 0 |
| 1 | 240 | 0 |
| 3 | 60 | 180 |
| 3 | 300 | 180 |

*Figure 0-21* **The alternate E-COSY pulse sequence with appropriate phase cycle and weighting factors. For psi = 180, the weights may be replaced by negative values (i.e. subtract FID’s).**

The evolution of the density matrix for this E-COSY sequence is depicted below in terms of Hilbert space propagators.

***Alternative E-COSY Sequence in Terms of Hilbert Space Propagators***



Density matrix evolution through this E-COSY sequence is given in the following set of equations.

1. C. Griesinger, O.W. Sørensen, and R.R. Ernst (1986), *J. Chem. Phys.*, **85**, 6837-6851, “Correlation of connected transitions by two-dimensional NMR spectroscopy”. See FIG. 4 on page 6839.

$$\sigma_1 = U_p(x)\sigma_{eq}U_p^{-1}(x) \tag{EQ 5}$$

$$\sigma_2(t_1) = U_d(t_1)\sigma_1 U_d^{-1}(t_1) \tag{EQ 6}$$

$$\sigma_3(t_1, \phi) = U_p(y, \phi)\sigma_2(t_1)U_p^{-1}(y, \phi) \tag{EQ 7}$$

$$\sigma(t_1, t_2, \phi) = U_d(t_2)\sigma_3(t_1, \phi)U_d^{-1}(t_2) \tag{EQ 8}$$

The FID is then computed by performing trace operations with a detection operator, $F_-$, which is also phase cycled.

$$FID(t_1, t_2, \phi, \psi) = Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} \tag{8-1}$$

The last step is that we sum the FID's over the phase cycle

$$FID(t_1, t_2) = \sum_{\phi, \psi}^{\text{cycle}} FID(t_1, t_2, \phi, \psi) \tag{8-2}$$

Unlike the previous example, the step implementing the phase cycle is virtually independent from the steps involving the $t_1$ and $t_2$ time increments. Use of superoperator will not be done but we will still isolate the phase cycle. The first mathematical adjustment will be the realization that the detection phase cycle simply adds (0) or subtracts (180) the FID, and that we can remove the psi angle from the detection operator.

$$FID(t_1, t_2, \phi, \psi) = Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} = (-1)^{\psi/\pi}Tr\{F_-\sigma(t_1, t_2, \phi)\} \quad .$$

Expanding this result based on the previous equations produces

$$FID(t_1, t_2, \phi, \psi) = (-1)^{\psi/\pi}Tr\left\{F_-U_d(t_2)\sigma_3(t_1, \phi)U_d^{-1}(t_2)\right\} \quad .$$

$$FID(t_1, t_2, \phi, \psi) = (-1)^{\psi/\pi}Tr\{F_-U_d(t_2)U_p(y, \phi)\sigma_2(t_1)U_p^{-1}(y, \phi)U_d^{-1}(t_2)\}$$

For the E-COSY we must sum over the phase cycle, weighting each by an appropriate amount.

$$FID(t_1, t_2) = \sum_{\phi, \psi}^{\text{cycle}} (-1)^{\psi/\pi}Tr\{F_-U_d(t_2)U_p(y, \phi)\sigma_2(t_1)U_p^{-1}(y, \phi)U_d^{-1}(t_2)\}$$

$$FID(t_1, t_2) = Tr\left\{F_-U_d(t_2)\sum_{\phi, \psi}^{\text{cycle}} W_\phi(-1)^{\psi/\pi}[U_p(y, \phi)\sigma_2(t_1)U_p^{-1}(y, \phi)]U_d^{-1}(t_2)\right\} \tag{8-3}$$

The factor (-1) in this scheme can be included in the weighting factor and our equation simplifies to

$$FID(t_1, t_2) = Tr\left\{ F_{-} U_d(t_2) \sum_{\phi}^{\text{cycle}} W_{\phi}[U_p(y, \phi)\sigma_2(t_1)U_p^{-1}(y, \phi)]U_d^{-1}(t_2) \right\} \quad , \tag{8-4}$$

and the necessary weights, $W_{\phi}$, are the +/- those previously given; their sign depends upon the angle phi (or correspondingly the detector angle). The simulation will be performed in this manner.

***Program Outline for the Alternative E-COSY Sequence***

| **Step** | | **Density Matrix** |
|---|---|---|
| **1.** | ***Form Equilibrium Density Matrix*** | $\sigma_{eq}$ |
| **2.** | ***Apply the 1st 90 x-pulse*** | $\sigma_1$ |
| **3.** | ***Evolve for $t_1$ time*** | $\sigma_2(t_1)$ |
| **4.** | ***Apply phi pulse, E-COSY Phase Cycle*** | $\sigma_3(t_1)$ |
| **5.** | ***Acquire FID for this $t_1$ time*** | $\sigma(t_1, t_2)$ |
| **6.** | ***Repeat Steps 3-6 for each $t_1$ time*** | |

Note that this scheme is equivalent to the outline used in the previous example. The difference lies in the implementation of Step 4. Previously, a superoperator was used in implementing the E-COSY phase cycle. Now, after t1 evolution, the system will be evolved through y-pulses of differing angles and the results scaled by a weighting factor then summed.

## 2.4.2 Program

```
/* ecosy2.cc *******************************************************************-*-c++-*-
**                                                                              **
**                    Example Program for the GAMMA Library                     **
**                                                                              **
** This program simulates either an E-COSY experiment with                      **
** complete phase cycling. The pulsesare taken to be ideal                      **
** and the effects of relaxation are not considered.                            **
**                                                                              **
** See: C. Griesinger, O.W. Sorensen, and R.R. Ernst, JCP,                      **
** 85(12), 6837-6852, (1986), "Correlation of Connected                         **
** Transitions by Two-Dimensional NMR Spectroscopy". The                        **
** required pulse sequence is found on page 6839 of this                        **
** reference.                                                                   **
**                                                                              **
** Note: This program treats only homonuclear spin systems.                     **
**                                                                              **
********************************************************************************* */

#include "gamma.h"
//                              Define Constants
 const P_cycl = 5;                                    // Phase cycle length
 const double P_mix[5] = {0,60,120,240,300};          // Pulse phase cycle, E-COSY
 const double P_det[5] = {4,-3,1,1,-3};               // Detector phase cycle, E-COSY
//                              Begin Program
main (int argc, char* argv[])

 {
 cout << "\n\t\tAlternate Simulation of E-COSY\n";
 cout << "\t (TPPI Mode, Hard Pulses, No Relaxation)\n";
                                Read in Parameters
 String filename;                                     // Name of spin system file
 query_parameter(argc, argv, 1,                       // Get filename from command
        "\n\tSpin system filename? ", filename);      // line or ask for it
 spin_system sys;                                     // Declare spin system sys
 sys.read(filename);                                  // Read system from filename
 int t1pts, t2pts;
 query_parameter(argc, argv, 2,                       // Get number FID of points
      "\n\tAcquisition Size? ", t2pts);
 t1pts = 2*t2pts;                                     // Set t1 size for TPPI
 String J;
 query_parameter(argc, argv, 3,                       // Weak or strong coupling
        "\n\tWeak or strong coupling (w/s)? ", J);
 double offset = query_offset(sys, 0, 1);             // Ask for an offset frequency
 double NyqF = query_Nyquist(sys,0);                  // Choose a Nyquist frequency
 double t2dt = 1.0/(2.0*NyqF);                        // Dwell time, quadrature
 double t1dt = t2dt/2.0;                              // t1 time increment, TPPI
```

```
                     Set Up Operators, Superoperator
gen_op H;                                             // Set Hamilitonian for
 if(J == "w")                                         // Strong or weak coupling
       H = Hcs(sys) + HJw(sys);
 else
       H = Hcs(sys) + HJ(sys);
 gen_op Ud1 = Rz(sys,+90)*prop(H,t1dt);               // Delay t1 + TPPI propagator
 gen_op D = Fm(sys);                                  // Detector to F-
 gen_op Upx = Ixypuls_U(sys, 0., 90.);                // Propagator for 90-x pulse
 gen_op Uphi[P_cycl], Uphiy;                          // Propagators for phi pulses
 for(int j=0; j<P_cycl; j++ )
      {
      Uphiy = Ixypuls_U(sys,90.,P_mix[j]);            // Generate phi-y pulse props
      Uphi[j] = Uphiy;                                // Store this phi-y pulse props

 gen_op sigma3, sigman;
//                        Apply Pulse Sequence
File ecosy;                                           // Declare and open file
 ecosy.open("ecosy.dat",
 io_writeonly, a_create);
 block_1D t2BLK(t2pts);                               // Set 1D block for output
 gen_op sigma0 = sigma_eq(sys);                       // Set density matrix equilib.
 gen_op sigma1 = evolve(sigma0, Upx);                 // Apply first (PI/2)x pulse
 gen_op sigma2 = sigma1;                              // Initial sigma2 (t1 = 0)
 for(int t1=0; t1<t1pts; t1++)                        // Loop over all t1 increments
       {
       sigma3 = sigman;                               // Set sigma3 to NULL
       for(int i=0; i<P_cycl; i++ )                   //apply E-COSY phase cycle
       sigma3 += P_det[i]*evolve(sigma2,Uphi[i]);//
       FID(sigma3,D,H,t2dt,t2pts,t2BLK);              //acquisition
       Felix(ecosy, t2BLK);                           //output block: Felix
       evolve_ip(sigma2, Ud1);                        //evolution next t1 + TPPI
       }
 ecosy.close();                                       // Close file
 double Om = sys.Omega();                             // Spectrometer frequency
 Felix2D_params(cout, Om, 2.0*NyqF,
                                  2*t2pts, offset);   // Output Felix parameters
 cout << "\n\n";                                      // Keep screen nice
 }
```

### 2.4.3 Discussion

**Define Constants:** In this program the E-COSY phase cycle is explicitly defined; A 5 step cycle with associated phases and weights are stored in accordance with Figure 0-21.

**Read in Parameters:** As in prior simulations the spin system is read in from a disk file, the name may be supplied directly when running the program. The user is also asked for the acquisition size, whether to use a strong or weak coupling Hamiltonian, any desired offset, and a Nyquist frequency. Note that because TPPI will be used for phase sensitivity along the t1/f1 axis, twice as many points are taken along this axis that along the t2 axis, and the t1 incrementation time is half of the dwell time.

**Set up Operators, Superoperator:** The Hamiltonian is first formulated for either strong or weak coupling depending upon the input choice. Next, the $t_1$ time increment propagator is formulated. Here, in order to produce an acquisition with TPPI, a 90 degree rotation about the z-axis is added so that each incremented step includes the TPPI phase shift. The detector is set to F- in the next line. A propagators for the 90x pulse is now computed, followed by generation of propagators for all five y-pulses of angle phi. Operators are declared, the first for a working density matrix and the second for a null operator.

**Apply Pulse Sequence:** A file name ecosy is created and opened. A 1D data block called t2BLK is constructed to temporarily store the FID. Following the pulse sequence diagram, the initial density matrix is set to equilibrium. The next step is to apply a 90 degree pulse (without phase cycling) along the x-axis. This is evolved next by the propagator which accounts for $t_1$ incrementation as well as TPPI phase adjustments, but it is not done on the first step through the $t_1$ increments, at $t_1$=0. Rather, this step is done at the end of the loop. The next step is evolution under the final pulse of angle phi along the y-axis. The E-COSY phase cycle is implemented at this step as discussed previously. This density matrix, sigma3, is then used to simulate an FID which is subsequently sent to the file ecosy.dat in Felix format. Again, the last step of the loop is actually the earlier t1 incrementation. The last program steps close the file, get the spectrometer frequency, output Felix parameters necessary for spectral workup, and clear the screen.

## 2.4.4 Spin System

For this example we will first treat a 4 spin proton system in order verify that the simulation produces correct results. We shall use a system derived from one used in a previous simulation so that we can readily see how to interpret the E-COSY cross peak pattern.

***Initial 3-Spin Proton System for Alternate E-COSY Verification***

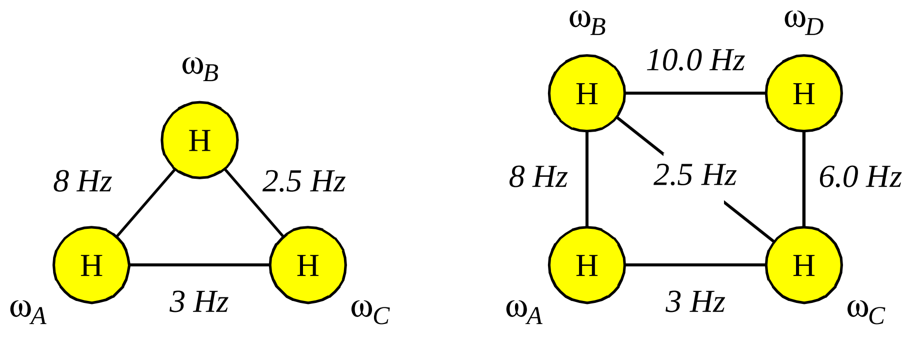


***Figure 0-22*** **Simple 3 and 4 spin protons systems to be utilized in verification of the alternate E-COSY simulation program. The three spin system is identical to that used in the previous example and the 4 spin system is directly derived from it.**

ecosy2.sys

| | | |
|---|---|---|
| **SysName** | **(2) : ECOSY2** | **- Name of the Spin System** |
| **NSpins** | **(0) : 4** | **- Number of Spins in the System** |
| **Iso(0)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(1)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(2)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(2)** | **(2) : 1H** | **- Spin Isotope Type** |
| **PPM(0)** | **(1) : -.1** | **- Chemical Shifts in PPM** |
| **PPM(1)** | **(1) : 0.0** | **- Chemical Shifts in PPM** |
| **PPM(2)** | **(1) : .1** | **- Chemical Shifts in PPM** |
| **J(0,1)** | **(1) : 8.0** | **- Coupling Constants in Hz** |
| **J(0,2)** | **(1) : 3.0** | **- Coupling Constants in Hz** |
| **J(1,2)** | **(1) : 2.5** | **- Coupling Constants in Hz** |
| **Omega** | **(1) : 500** | **- 1H Spectrometer Frequency in MHz** |

## 2.4.5 Workup

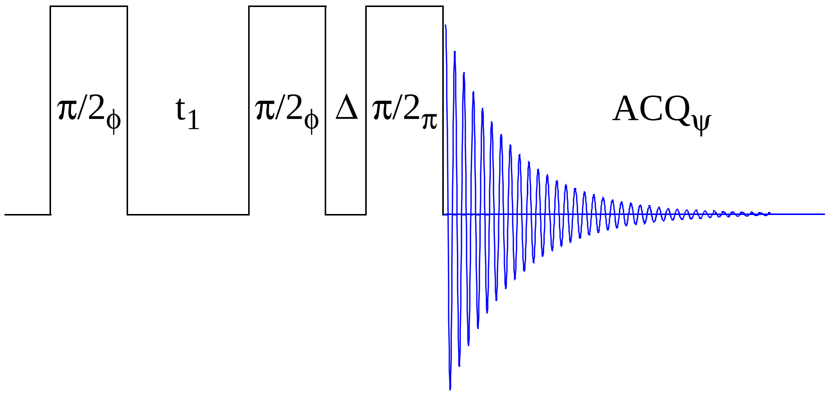
The 2D-data sets produced from this simulation is processed by the identical procedure detailed in the previous example, using an acquisition size of 512 points and the provided macro ecosy1.mac.

## 2.5 Complimentary E-COSY with Superoperators

### 2.5.1 Description

As was mentioned at the beginning of this chapter, complimentary E- COSY experiments involve only a change in the weighting factors relative to the cycled pulse phases.

***Complimentary E-COSY & E-COSY Pulse Sequence with Phase Cycle***



| W | $\phi_{CE}$ | $\phi_C$ | $\psi$ |
|---|---|---|---|
| 4 | 180 | 0 | 0 |
| 1 | 300 | 120 | 0 |
| 1 | 60 | 240 | 0 |
| 3 | 240 | 60 | 180 |
| 3 | 120 | 300 | 180 |

***Figure 0-23*** **The E-COSY sequence used in the first example. The complementary E-COSY associates the same weights, W, with different pulse phases $\phi$. Here $\phi_{CE}$ is used for complementary E-COSY and $\phi_E$ for E-COSY.**

We shall not review the mathematical formalism for this simulation as it can be found in the previous example. Rather, we shall make only minor modifications to the GAMMA program implementing E-COSY in the first example to produce a program which will generate either the E-COSY or complimentary E-COSY sequence. The focus will be on understanding the difference between E-COSY and complimentary E-COSY experiments.

## 2.5.2 Program

```
/* ecosy3.cc ******************************************************************-*-c++-*-
**                                                                                  **
**                      Example Program for the GAMMA Library                       **
**                                                                                  **
** This program simulates either an E-COSY or complimentary                         **
** E-COSY experiment with complete phase cycling. The pulses                        **
** are taken to be ideal and the effects of relaxation are                          **
** not considered.
**
**                                                                                  **
** See: C. Griesinger, O.W. Sorensen, and R.R. Ernst, JMR,                          **
** 75, 474-492, (1987), "Practical Aspects of the E.COSY                            **
** Technique. Measurement of Scalar Spin-Spin Coupling                              **
**
Constants in Peptides". The required phase cycle is                                 **
**
found on page 477 of this reference.                                                **
**                                                                                  **
** Note: This program treats only homonuclear spin systems.                         **
**
Superoperators are used to implement the phase cycle                                **
** which speeds up the computation for small systems                                **
** but this program is cumbersome for large systems.                                **
**                                                                                  **
*****************************************************************                   */

#include "gamma.h"
//                              Define Constants

const P_cycl = 12;                                  // Phase cycle length
 const double P_mix[12] =                           // Pulse phase cycle, E-COSY

{0,60,0,60,0,60,0,300,120,300,240,300};
 const double PC_mix[12] =                          // Pulse phase cycle, CE-COSY

{180,240,180,240,180,240,180,120,300,120,60,120};
//                              Begin Program

main (int argc, char* argv[])
{
 cout << "\n\tSimulation of E-COSY and Complimentary E-COSY\n";
 cout << "\t (TPPI Mode, Hard Pulses, No Relaxation)\n";
//                              Read in Parameters

String filename;                                    // Name of spin system file
 query_parameter(argc, argv, 1,                     // Get filename from command
 "\n\tSpin system filename? ", filename);           // line or ask for it
 spin_system sys;                                   // Declare spin system sys
 sys.read(filename);                                // Read system from filename
 int t1pts, t2pts;
 query_parameter(argc, argv, 2,                     // Get number FID of points
       "\n\tAcquisition Size? ", t2pts);
 t1pts = 2*t2pts;                                   // Set t1 size for TPPI
 String J;
 query_parameter(argc, argv, 3,                     // Weak or strong coupling
 "\n\tWeak or strong coupling (w/s)? ", J);
 String CE;
 query_parameter(argc, argv, 4,                     // Type of E-COSY desired
 "\n\tComplimentary E-COSY or E-COSY (c/e)? ",CE);
 double offset = query_offset(sys, 0, 1);           // Ask for an offset frequency
 double NyqF = query_Nyquist(sys,0);                // Choose a Nyquist frequency
 double t2dt = 1.0/(2.0*NyqF);                      // Dwell time, quadrature
 double t1dt = t2dt/2.0;                            // t1 time increment, TPPI
                        Set Up Operators, Superoperator
gen_op H;                                           // Set Hamilitonian for
 if(J == "w")                                       // Strong or weak coupling
   H = Hcs(sys) + HJw(sys);
 else
   H = Hcs(sys) + HJ(sys);
 gen_op Ud1 = Rz(sys,+90)*prop(H,t1dt);             // Delay t1 + TPPI propagator
 gen_op D = Fm(sys);                                // Detector to F-
 gen_op Upx = Ixypuls_U(sys,0.,90.);                // Propagator for x pulse
 gen_op Upmx = Ixypuls_U(sys,0.,-90.);              // Propagator for -x pulse
 gen_op U_mix;                                      // Temporary mixing propagator
 super_op G_mix;                                    // Phase cycle superoperator
 gen_op sigma3;
//              Construct Mixing, Phase Cycle Superoperator
double conv = acos(-1)/180;
 double P_det = 0;
 if(CE == "c")                                      // This for comp. E-COSY
   cout << "\n\tComplimentary E-COSY Phase Cycle\n";
 else
   cout << "\n\tE-COSY Phase Cycle\n";
 cout << "\n\t# scan                         betareference\n";
 for(int i=0; i<P_cycl; i++ )
      {
      U_mix = Rz(sys,-P_det);                       // Detector phase cycle
      U_mix *= Upmx;                                // 3rd 90 Pulse (-x)
      if (CE == "c")                                // This for comp. E-COSY
          {
          cout << "\t"<< i <<"\t" << PC_mix[i];     // Output comp. E-COSY phase
          U_mix *= Rz(sys,+PC_mix[i]);              // Phase shift pulses 1 & 2
          }
      else                                          // This for E-COSY
          {
          cout << "\t"<< i << "\t"<< P_mix[i];      // Output E-COSY phase
          U_mix *= Rz(sys,+P_mix[i]);               // Phase shift pulses 1 & 2
          }
      cout << "\t" <<P_det<< "\n";
      U_mix *= Upx;                              // 2nd 90 Pulse
      U_mix.Op_base(H);                          // Put in eigenbasis of Ho
      G_mix += U_transform(U_mix);               // Add to U transform superop
      P_det = acos(-cos(P_det*conv))/conv;       // Adjust detector phase
      }
//                    Apply Pulse Sequence
File ecosy;                                      // Declare and open file
 ecosy.open("ecosy.dat",
 io_writeonly, a_create);
 block_1D t2BLK(t2pts);                          // Set 1D block for output
 gen_op sigma0 = sigma_eq(sys);                  // Set density matrix equilib.
 gen_op sigma1 = evolve(sigma0, Upx);            // Apply first (PI/2)x pulse
 gen_op sigma2 = sigma1;                         // Initial sigma2 (t1 = 0)
 for (int t1=0; t1<t1pts; t1++)                  // Loop over all t1 increments
      {
      sigma3 = evolve(sigma2, G_mix);            //superop phase cycle
      FID(sigma3,D,H,t2dt,t2pts,t2BLK);          //acquisition
      Felix(ecosy, t2BLK);                       //output block: Felix
      evolve_ip(sigma2, Ud1);                    //evolution next t1
      }
 ecosy.close();                                  // Close file
 double Om = sys.Omega();                        // Spectrometer frequency
 Felix2D_params(cout, Om, 2.0*NyqF,
                          2*t2pts, offset);      // Output Felix parameters
 cout << "\n\n";                                 // Keep screen nice
 }
```

### 2.5.3 Discussion

**Define Constants:** Prior to entering the program, the size of the phase cycle (12) and the phase angles for the pulses are set for both the E-COSY and complimentary E-COSY experiment.

**Read in Parameters:** As in prior simulations the spin system is read in from a disk file, the name may be supplied directly when running the program. The user is also asked for the acquisition size, whether to use a strong or weak coupling Hamiltonian, any desired offset, and a Nyquist frequency. In the program the user is also asked whether E-COSY or complimentary E-COSY is desired. Note that because TPPI will be used for phase sensitivity along the t1/f1 axis, twice as many points are taken along this axis than along the t2 axis, and the t1 incrementation time is half of the dwell time.

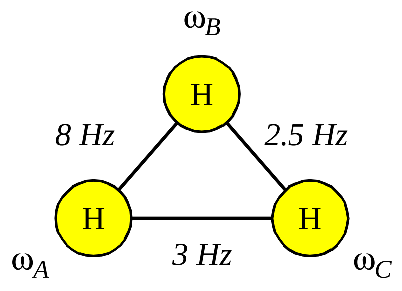
**Set up Operators, Superoperator:** The Hamiltonian is first formulated for either strong or weak coupling depending upon the input choice. Next, the $t_1$ time increment propagator is formulated. Here, in order to produce an acquisition with TPPI, a 90 degree rotation about the z-axis is added so that each incremented step includes the TPPI phase shift. The detector is set to F- in the next line. Propagators for the two 90 pulses are now computed, the first along x and the second along -x. A temporary operator is declared for used in summing over the phase cycle. The next line declares a superoperator which will ultimately account for the phase cycle. Finally, a working density matrix is specified.

**Construct Mixing, Phase Cycle Superoperator:** As outline mathematically in the first example, the entire phase cycle can be removed from the t1 and t2 looping by use of a superoperator. In this section, the phase cycle loop is applied and the appropriate superoperator constructed, now the cycle depends upon whether E-COSY or complimentary E-COSY has been specified. The loop goes over the twelve steps of the cycle (defined in the constants section) and at each step the phases are output to the terminal. Initially the detector phase is set to zero. The propagators for each step are then multiplied together in the opposite order of the pulse sequence due to the ordering of the unary *= step. At the end of each cycle, the unitary transform superoperator is summed to account for the mixing propagator. The last step in this section adjusts the detector phase to +/- 180.

**Apply Pulse Sequence:** Since the superoperator intrinsically contains the phase cycling, the pulse sequence application is quite simple. A file name ecosy is created and opened. A 1D data block called t2BLK is constructed to temporarily store the FID. Following the pulse sequence diagram, the initial density matrix is set to equilibrium. The next step is to apply a 90 degree pulse (without phase cycling) along the x-axis. This is evolved next by the propagator which accounts for $t_1$ incrementation as well as TPPI phase adjustments, but it is not done on the first step through the $t_1$ increments, at $t_1$=0. Rather, this step is done at the end of the loop. The next step is evolution under the superoperator which contains all the phase cycling. This density matrix, sigma3, is then used to simulate an FID which is subsequently sent to the file ecosy.dat in Felix format. Again, the last step of the loop is actually the earlier t1 incrementation. The last program step simply closes the file.

### 2.5.4 Spin System

For this example we will use the same three spin system as in our first example.



The E-COSY simulation will be run on a number of 3- and 4- spin system which contain protons and coupling constants for this system. These are given below. Only two of the systems are given in full, the others are derived from these two.

**ecosy1a.sys**

| | | |
|---|---|---|
| **SysName** | **(2) : ECOSY1A** | **- Name of the Spin System** |
| **NSpins** | **(0) : 3** | **- Number of Spins in the System** |
| **Iso(0)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(1)** | **(2) : 1H** | **- Spin Isotope Type** |
| **Iso(2)** | **(2) : 1H** | **- Spin Isotope Type** |
| **PPM(0)** | **(1) : -.1** | **- Chemical Shifts in PPM** |
| **PPM(1)** | **(1) : 0.0** | **- Chemical Shifts in PPM** |
| **PPM(2)** | **(1) : .1** | **- Chemical Shifts in PPM** |
| **J(0,1)** | **(1) : 8.0** | **- Coupling Constants in Hz** |
| **J(0,2)** | **(1) : 3.0** | **- Coupling Constants in Hz** |
| **J(1,2)** | **(1) : 2.5** | **- Coupling Constants in Hz** |
| **Omega** | **(1) : 500** | **- 1H Spectrometer Frequency in MHz** |

### 2.5.5 Workup

The 2D-data sets produced from this simulation is processed by the identical procedure detailed in the previous example, using an acquisition size of 512 points and the provided macro ecosy1.mac.

## 2.5.6 Results

***Simulated Complimentary E-COSY and E-COSY Spectra***

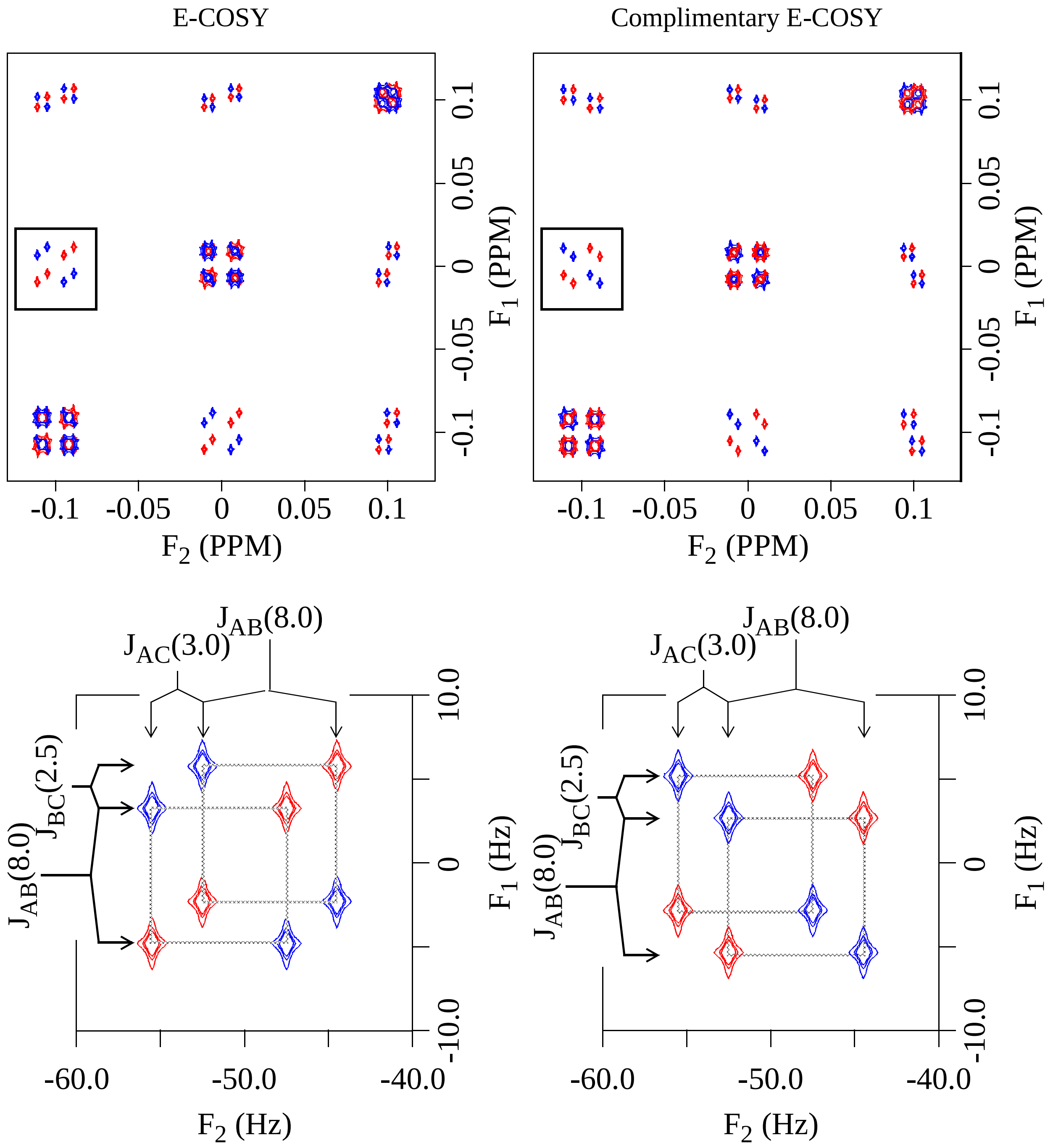


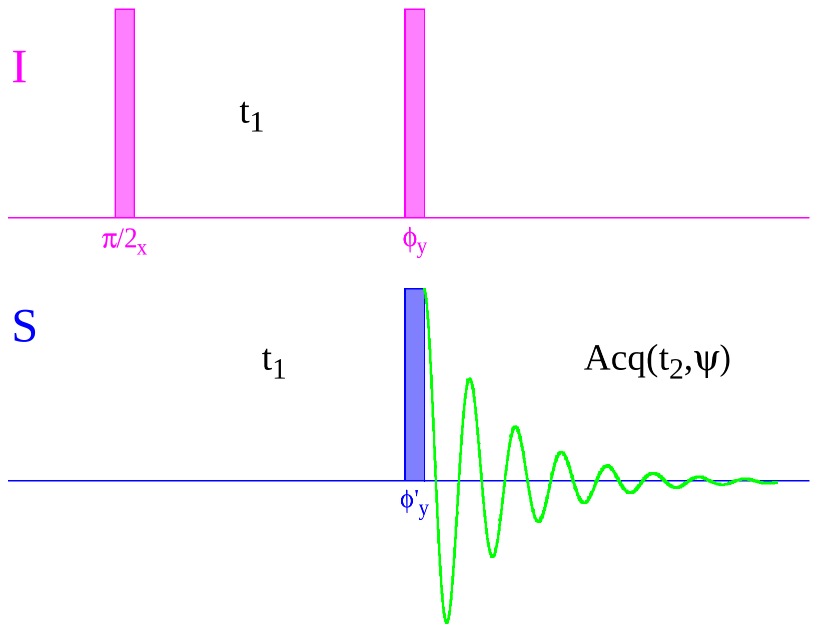
***Figure 0-24*** **Comparison of the E-COSY and complimentary E-COSY sequences on a 3-spin proton system. The information content is equivalent, the shifts due to the passive spin(s) are in opposite directions.**

## 2.6 Heteronuclear E-COSY

### 2.6.1 Description

We shall employ a variant of the pulse sequence used in example 2 of this section to simulation an E-COSY experiment on a heteronuclear spin system. Employment of the sequence shown below is discussed in the literature.[1]

***Heteronuclear E-COSY Pulse Sequence and Phase Cycle***



| W | ϕ | ψ |
|---|---|---|
| 14.9 | 0 | 180 |
| 2 | 90 | 0 |
| 1.1 | 150 | 180 |
| 1.1 | 210 | 0 |
| 2 | 270 | 180 |
| 14.9 | 330 | 0 |

***Figure 0-25*** **: The homonuclear E-COSY pulse sequence with appropriate phase cycle and weighting factors. For psi = 180, the weights may be replaced by negative values (i.e. subtract FID's).**

In this experiment, the angles ϕ and ϕ' have the same phase cycles but they cycle independently. The angles and relative weights are given by the following equations

$$\phi_j = j(\pi/N) + \delta_{N,K}\pi/(2K) \qquad j = 0, 1, \ldots, 2N-1\ . \tag{8-5}$$

$$W_j = \frac{(-1)^{j+1}}{\sin^2(\beta_j/2)}\ . \tag{8-6}$$

the latter used when N=K. The values in the figure utilized N=K=3. The evolution of the density matrix for the heteronuclear E-COSY sequence is virtually identical with the treatment of the homonuclear E-COSY experiment given in the second example of this Chapter (Alternative E-

1. C. Griesinger, O.W. Sørensen, and R.R. Ernst (1986), *J. Chem. Phys.*, **85**, 6837-6851, "Correlation of connected transitions by two-dimensional NMR spectroscopy". See FIG. 4 on page 6839 as well as equation (43) on page 6847.

COSY pulse sequence). The changes are only in that the pulses are selective for specific isotopes and the last pulse is applied to two different isotopes with two different phases. As such we shall here provide only an outline for the simulation program.

***Program Outline for the Homonuclear E-COSY Sequence***

| **Step** | **Density Matrix** |
|---|---|
| **1.** ***Form Equilibrium Density Matrix*** | $\sigma_{eq}$ |
| **2.** ***Apply the 1st 90 x-pulse on I*** | $\sigma_1$ |
| **3.** ***Evolve for $t_1$ time*** | $\sigma_2(t_1)$ |
| **4.** ***Apply phi pulse on S*** | $\sigma_3(t_1)$ |
| **5.** ***Apply phi pulse on I, E-COSY Phase Cycle on I*** | $\sigma(t_1, t_2)$ |
| **6.** ***Repeat Steps 4 and 5, E-COSY Phase Cycle on S*** | |
| **7.** ***Acquire FID for this $t_1$ time*** | $\sigma(t_1, t_2)$ |
| **8.** ***Repeat Steps 3-7 for each $t_1$ time*** | |

## 2.6.2 Program

```
/* ecosy4.cc ***********************************************-*-c++-     *-
**                                                                     **
** Example Program for the GAMMA Library                               **
**                                                                     **
** This program simulates either an E-COSY experiment with             **
** complete phase cycling. The pulses
are taken to be ideal                                                  **
** and the effects of relaxation are not considered.                   **
**                                                                     **
** See: C. Griesinger, O.W. Sorensen, and R.R. Ernst, JCP,             **
** 85(12), 6837-6852, (1986), "Correlation of Connected                **
** Transitions by Two-Dimensional NMR Spectroscopy". The required      **
** required pulse sequence is found on page 6839 of this reference     **
**                                                                     **
** Note: This program treats heteronuclear spin systems.               **
**                                                                     **
*************************************************************************/


#include "gamma.h"

void setup_2D(spin_system& sys, String& S, String& I, int& isoS,
          int& isoI, double& offS, double& offI, double& NyqFS,
                        double& NyqFI, int& t2pts, int& t1pts=-1)
{
 isoS = query_isotope(sys, S,                       // Choose an isotope to detect
                 "\n\tIsotope type for detection? ");
 isoI = query_isotope(sys, I,                       // Choose an 2nd channel isotope
               "\n\tIsotope type for heteroatom? ");
 if(isoS == isoI)
     {
     cout << "\n\n\tSorry, The Two Spin Types Are Not"
            << " Allowed to Be The Same\n\n";
     exit(-1);
     }
 offS = query_offset(sys, isoS);                    // Ask for an S offset frequency
 NyqFS = query_Nyquist(sys, isoS);                  // Choose an S Nyquist frequency
 offI = query_offset(sys, isoI);                    // Ask for an I offset frequency
 NyqFI = query_Nyquist(sys, isoI);                  // Choose an I Nyquist frequency
 cout << "\n\tAcquisition Size? ";                  // Get number FID of points
 cin >> t2pts;
 if(t1pts != -1)
     {
     cout << "\n\tNumber t1 Increments? ";          // Get number of t1 points
     cin >> t1pts;
     }
 return;
 }

//                          Begin Program
main (int argc, char* argv[])

 {
 cout << "\n\t\tHeteronuclear E-COSY Simulation\n";
 cout << "\t (TPPI Mode, Hard Pulses, No Relaxation)\n";
//                       Read in Parameters
spin_system sys;
 query_sys(argc, argv, 1, sys);                     // Get the spin system
 if(sys.homonuclear())                              // Only heteronuclear allowed
     {
     cout << "\n\n\tPlease Run One of the Homonuclear"
            << " E-COSY Example Programs for This System\n\n";
     exit(-1);
     }
 String IsoI, IsoS;                                 // Two isotope types
 int isoS, isoI, t1pts, t2pts;
 double offS, offI, NyqFS, NyqFI;
 setup_2D(sys, IsoS, IsoI, isoS, isoI,
 offS, offI, NyqFS, NyqFI, t2pts);
 double t2dt = 1.0/(2.0*NyqFS);                     // Dwell time, quadrature
 double t1dt = 1.0/(4.0*NyqFI);                     // t1 time increment, TPPI
 t1pts = 2*t2pts;                                   // Set t1 size for TPPI
 String J;
 query_parameter(argc, argv, 2,                     // Weak or strong coupling
 "\n\tWeak or strong coupling (w/s)? ", J);
//             Set Up The Heteronuclear E-COSY Phase Cycle
int P_cycl = 6;                                     // Phase cycle length
 double P_mix[P_cycl];                              // Pulse phase cycle angles
 double P_det[P_cycl];                              // Weighting factors
 double sinBo2, sign = 1.0;
 for(int k=0; k<=P_cycl; k++)
     {
     P_mix[k] = double(k)*60.0 + 30.0;
     sinBo2 = sin(P_mix[k]*PI/360.0);               // Angle is phi/2 in radians
     sign *= -1.0;
     P_det[k] = sign/(sinBo2*sinBo2);
     }
//                  Set Up Operators, Superoperator
 gen_op H;                                          // Set Hamilitonian for
 if(J == "w")                                       // Strong or weak coupling
   H = Hcs(sys) + HJw(sys);
 else
   H = Hcs(sys) + HJwh(sys);
 gen_op Ud1 = Rz(sys,IsoI,+90.0)*prop(H,t1dt);      // Delay t1 + TPPI prop
 gen_op D = Fm(sys, IsoS);                          // Detector to F-
 gen_op Upx = Ixypuls_U(sys, IsoI, 0., 90.);        // Prop for 90-x pulse
 gen_op UphiS[P_cycl], UphiI[P_cycl], Uphiy;        // Prop for phi pulses
for(int jj=0; jj<P_cycl; jj++ )
  {
  Uphiy=Ixypuls_U(sys,IsoS,90.,P_mix[jj]);          // Phi-y S pulse props
  UphiS[jj] = Uphiy;                                // Store pulse prop
  Uphiy=Ixypuls_U(sys,IsoI,90.,P_mix[jj]);          // Phi-y I pulse props
  UphiI[jj] = Uphiy;                                // Store pulse prop
  }
gen_op sigma3, sigman, sigmat;

//                                                  Apply Pulse Sequence

File ecosy;                                         // Declare and open file
ecosy.open("ecosy.dat",
io_writeonly, a_create);
block_1D t2BLK(t2pts);                              // Set 1D block for output
gen_op sigma0 = sigma_eq(sys);                      // Set density matrix equilib.
gen_op sigma1 = evolve(sigma0, Upx);                // Apply first (PI/2)x pulse
gen_op sigma2 = sigma1;                             // Initial sigma2 (t1 = 0)
for(int t1=0; t1<t1pts; t1++)                       // Loop over all t1 increments
  {
  sigma3 = sigman;                                  // Set sigma3 to NULL
  for(int i=0; i<P_cycl; i++ )                      //apply phase cycle I
    {
    sigmat = P_det[i]*evolve(sigma2,UphiI[i]);
    for(int j=0; j<P_cycl; j++ )                    //apply phase cycle S
      sigma3+=P_det[j]*evolve(sigmat,UphiS[j]);
    }
  FID(sigma3,D,H,t2dt,t2pts,t2BLK);                 //acquisition
  Felix(ecosy, t2BLK);                              // output block: Felix
  evolve_ip(sigma2, Ud1);                           //evolution next t1 + TPPI
  }
ecosy.close();                                      // Close file
double OmS = sys.Omega(IsoS);                       // Spectrometer frequency
double OmI = sys.Omega(IsoI);                       // Spectrometer frequency
Felix2D_params(cout,OmS,2.0*NyqFS,2*t2pts,       // Output Felix parameters
          offS,OmI,2.0*NyqFI,t1pts,offI,2);
cout << "\n\n";                                     // Keep screen nice
}
```

## 2.6.3 Discussion

**Initial Function:** A short function, setup_2D, has been placed at the start of this program which asks for parameters suited for a heteronuclear 2D experiment. The user is here asked for spin types, the acquisition size, any desired offsets, and Nyquist frequencies.

**Read in Parameters:** The spin system is read in from a disk file and then checked to insure it is indeed heteronuclear. The user is asked for all the appropriate parameters for the heteronuclear experiment in the previously set function, setup_2D. Delay times are set from the specified Nyquist frequencies. Note that because TPPI will be used for phase sensitivity along the t1/f1 axis, twice as many points are taken along this axis than along the t2 axis, and the t1 incrementation time is half of the dwell time. The user is then asked whether to use a strong or weak coupling Hamiltonian.

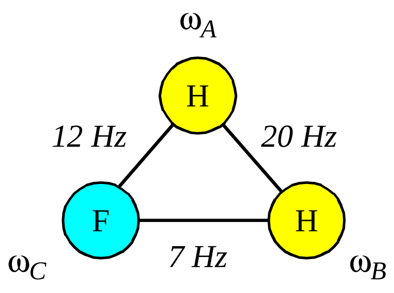
**Set Up The Heteronuclear E-COSY Phase Cycle:** Unlike the previous examples in this Chapter, the heteronuclear phase cycle and weighting functions are calculated. This is because the weights are not integer values so they are more accurately obtained by direct computation.

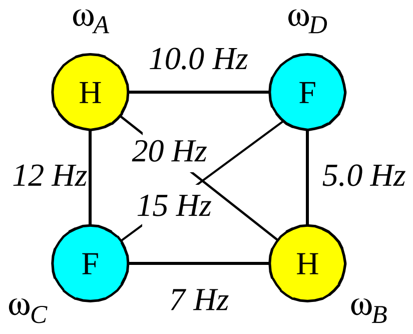
**Set up Operators, Superoperator:** The Hamiltonian is first formulated for either strong or weak coupling depending upon the input choice. Next, the $t_1$ time increment propagator is formulated. Here, in order to produce an acquisition with TPPI, a 90 degree rotation about the z-axis is added so that each incremented step includes the TPPI phase shift. The detector is set to F- in the next line. A propagators for the 90x pulse for spins of isotope type I is now computed, followed by generation of propagators for all y-pulses of angle phi. The phi pulses are performed on both I and S channels and may be applied with differing phases so the corresponding propagators are stored separately for each of the isotopes. Operators are declared for a working density matrix, a null operator, and a scratch operator.

**Apply Pulse Sequence:** A file name ecosy is created and opened. A 1D data block called t2BLK is constructed to temporarily store the FID. Following the pulse sequence diagram, the initial density matrix is set to equilibrium. The next step is to apply a 90 degree pulse (without phase cycling) along the x-axis. This is evolved next by the propagator which accounts for $t_1$ incrementation as well as TPPI phase adjustments, but it is not done on the first step through the $t_1$ increments, at $t_1$=0. Rather, this step is done at the end of the loop. The next step is evolution under the final pulse(s) of angle phi along the y-axis. The E-COSY phase cycle is implemented at this step as discussed previously. Because the phi pulse is applied to both channels and phase cycled independently on each a double loop runs through all combinations of phases for I and S. The resulting density matrix, sigma3, is then used to simulate an FID which is subsequently sent to the file ecosy.dat in Felix format. Again, the last step of the loop is actually the earlier t1 incrementation. The last program steps close the file, get the spectrometer frequencies, output Felix parameters necessary for spectral workup, and clear the screen.

## 2.6.4 Spin Systems

***Heteronuclear E-COSY Spin Systems for Example Runs***





**ecosy4a.sys**

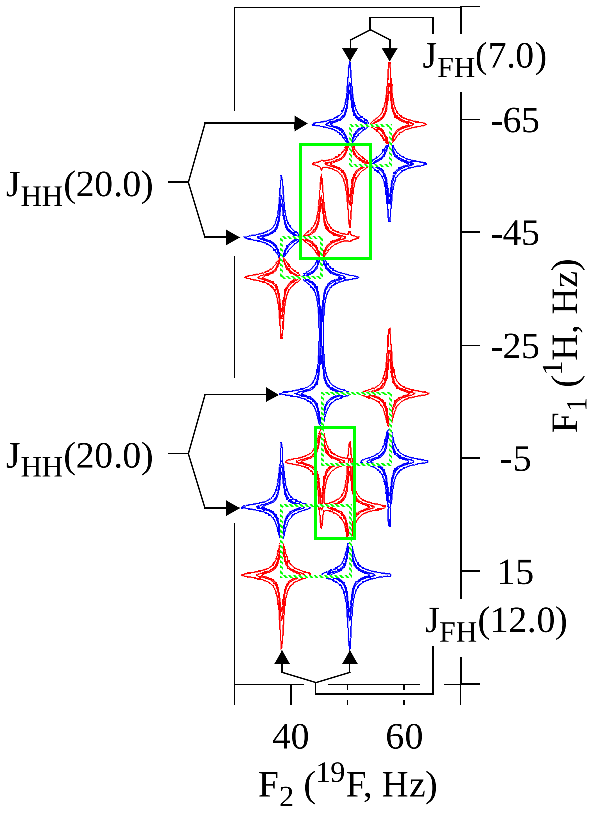
| | |
|---|---|
| **SysName (2) : ECOSY4A** | **- Name of the Spin System** |
| **NSpins (0) : 3** | **- Number of Spins in the System** |
| **Iso(0) (2) : 1H** | **- Spin Isotope Type** |
| **Iso(1) (2) : 1H** | **- Spin Isotope Type** |
| **Iso(2) (2) : 19F** | **- Spin Isotope Type** |
| **PPM(0) (1) : -.1** | **- Chemical Shifts in PPM** |
| **PPM(1) (1) : 0.0** | **- Chemical Shifts in PPM** |
| **PPM(2) (1) : .1** | **- Chemical Shifts in PPM** |
| **J(0,1) (1) : 20.0** | **- Coupling Constant JAB in Hz** |
| **J(0,2) (1) : 12.0** | **- Coupling Constant JAC in Hz** |
| **J(1,2) (1) : 7.0** | **- Coupling Constant JBC in Hz** |
| **Omega (1) : 500** | **- 1H Spectrometer Frequency in MHz** |

**ecosy4b.sys**

| | |
|---|---|
| **SysName (2) : ECOSY4B** | **- Name of the Spin System** |
| **NSpins (0) : 4** | **- Number of Spins in the System** |
| **Iso(0) (2) : 1H** | **- Spin Isotope Type** |
| **Iso(1) (2) : 1H** | **- Spin Isotope Type** |
| **Iso(2) (2) : 19F** | **- Spin Isotope Type** |
| **Iso(3) (2) : 19F** | **- Spin Isotope Type** |
| **PPM(0) (1) : -.1** | **- Chemical Shifts in PPM** |
| **PPM(1) (1) : 0.0** | **- Chemical Shifts in PPM** |
| **PPM(2) (1) : -.1** | **- Chemical Shifts in PPM** |
| **PPM(3) (1) : .1** | **- Chemical Shifts in PPM** |
| **J(0,1) (1) : 20.0** | **- Coupling Constant JAB in Hz** |
| **J(0,2) (1) : 12.0** | **- Coupling Constant JAC in Hz** |
| **J(0,3) (1) : 19.0** | **- Coupling Constant JAD in Hz** |
| **J(1,2) (1) : 7.0** | **- Coupling Constant JBC in Hz** |
| **J(1,3) (1) : 5.0** | **- Coupling Constant JBD in Hz** |
| **J(2,3) (1) : 15.0** | **- Coupling Constant JCD in Hz** |
| **Omega (1) : 500** | **- 1H Spectrometer Frequency in MHz** |

## 2.6.5 Results

**_Simulated Heteronuclear E-COSY Spectra_**



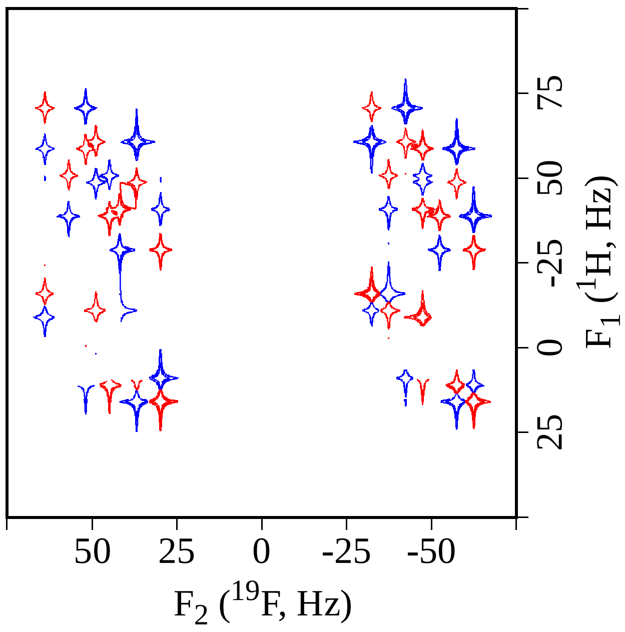


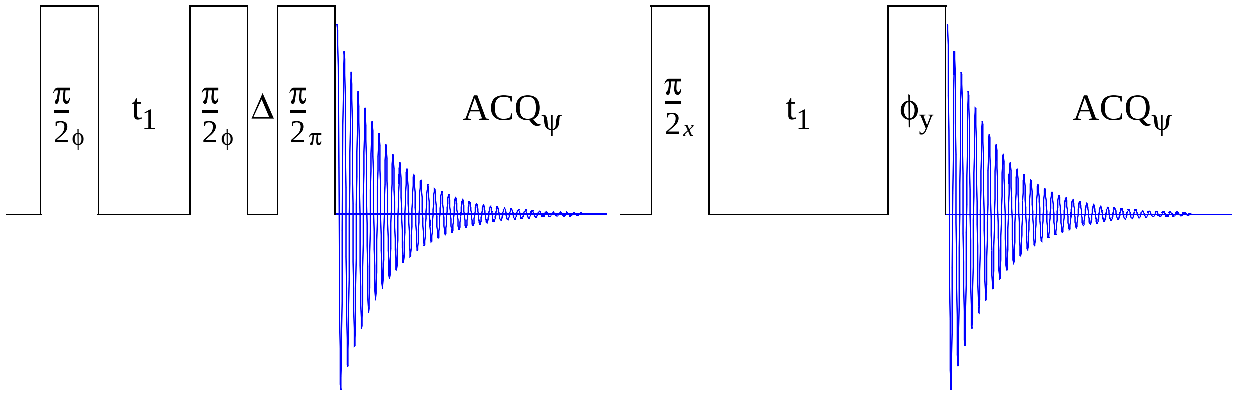
***Figure 0-26*** **Heteronuclear E-COSY simulations on a 3 and 4 spin system. The coupling constants are displayed on the 3 spin. The solid boxes show the shifts on the cross peaks due to the "passive" proton. The boxes are square with sides of length $J_{FH}$, where $J_{FH}$ is the scalar coupling to the passive spin.**

## 2.7 E-COSY Phase Cycling

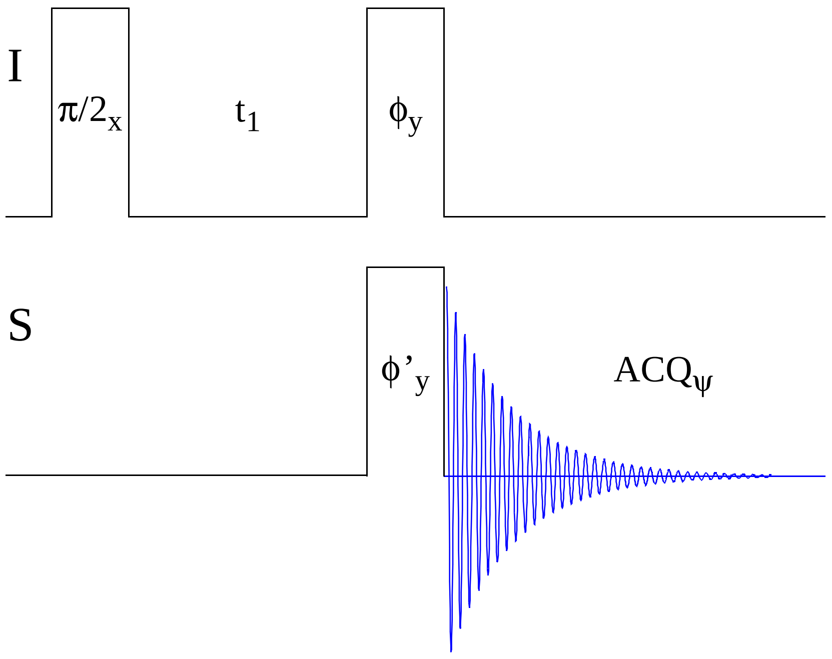
### 2.7.1 Description

In this chapter we have based our E-COSY simulations on a few basic pulse sequences which are found in the literature.

***Equivalent Homonuclear E-COSY Pulse Sequences***



***Heteronuclear E-COSY Pulse Sequence***



The most complicated aspect of implementing the E-COSY simulation is determining which phase cycle is appropriate. The program given here calculates the appropriate phases and weighting factors for the E-COSY experiment.

## 2.7.2 Program

```
/* ecosypc.cc *****************************************************************-*-c++-*-
**                                                                          **
** This program calculates the appropriate phase angles and weighting       **
** functions for an E-COSY experiment. The program runs interactive, asking **
** the user to supply the maximum coherence order expected (>2).            **
**                                                                          **
** See: C. Griesinger, O.W. Sorensen, and R.R. Ernst, JCP, 85(12), 6837-    **
** 6852, (1986), “Correlation of Connected Transitions by Two-Dimensional   **
** NMR Spectroscopy”.                                                       **
**                                                                          **
** Note: Although the parameters N, K, Bo, and B1 are included in the       **
** formulas which determine the phase cycle, they are restricted to the     **
** following values:                                                        **
**                                                                          **
**N = K = coherence order chosen                                            **
**B0 = B1 = 0                                                               **
**                                                                          **
** If different values are desired, the user will have                      **
** to make minor program modifications and re-compile.                      **
**                                                                          **
****************************************************************************** */

#include “gamma.h”
main (int argc, char* argv[])

 {
// Get the Coherence Order for the E-COSY Phase Cycle
 int ORD = 2;
 query_parameter(argc, argv, 1,                  // Coherence order for cycle
 “\n\tCoherence Order (>2): “, ORD);
 while(ORD < 3)
   {
   cout << “\n\tThe coherence order must be > 2”;
   cout << “\n\tCoherence Order (>2): “;
   cin >> ORD;
   }
 int P_cycl = 2*ORD;                             // E-COSY phase cycle length
 double B0 = 0.0;
 double B1 = 0.0;
 int N = ORD;
 int K = ORD;
//                 Cycle for Homonuclear E-COSY Phase Cycle
//    C. Griesinger, O.W. Sorensen and R.R. Ernst, JMR, 75, 474-492 (1987)
// phi = j * pi/N where j = [0, 2N-1]            [3], page 476
//    j
//           K
//           ---
//           \
// W = 0.5*B + /  B cos(p*phi ) - 0.5*B cos(K*phi )      [4], page 476
//  j      0  --- p        j       K        j
//           p
//     2                    2
// B = p /4 + B (p even) ; B = (p -1)/4 + B (p odd)      [1a,1b], page 475
//  p        0           p             1
//
//  comp
// W    = W                                              [4], page 476
//  j       j+N

 double phi_homo[P_cycl], phij;                     // Pulse phase cycle angles
 double W_homo[P_cycl];                             // Weighting factors
 double W_Chomo[P_cycl];                            // Weighting factors, compliment.
 double W, Bp, Cpfj;
 int eo;
 for(int j=0; j<P_cycl; j++)
      {
      phi_homo[j] = double(j)*180.0/double(N);      // The phase angles
      phij = phi_homo[j]*PI/180.0;                  // Convert phi to radians
      W = 0.5*B0;                                   // Begin weight factor
      eo = 1;
      for(int p=1; p<=K; p++)
          {
          eo *= -1;                                 // Track p even vs. p odd
          Bp = 0.25*double(p)*double(p);            // Determine Bp
          if(eo > 0)
            Bp += B0;
          else
            Bp += (B1 - 0.25);
          Cpfj = cos(double(p)*phij);               // Determine cos(p*phi)
          W += Bp*Cpfj;                             // Add to W
          }
      if(N==K)
          W -= 0.5*Bp*Cpfj;
      W_homo[j] = W;                                // Store weight factor
      }

 int jpn;
 double phijpn;
 for(j=0; j<P_cycl; j++)
   {
   jpn = j+N;
p  hijpn = double(jpn)*180.0/double(N);             // The phase angles
   phijpn *= (PI/180.0);                            // Convert phi to radians
   W = 0.5*B0;                                      // Begin weight factor
   eo = 1;
   for(int p=1; p<=K; p++)
     {
```

```
    eo *= -1;                                         // Track p even vs. p odd
    Bp = 0.25*double(p)*double(p);                    // Determine Bp
    if(eo > 0)
      Bp += B0;
    else
      Bp += (B1 - 0.25);
    Cpfj = cos(double(p)*phijpn);                     // Determine cos(p*phi)
    W += Bp*Cpfj;                                     // Add to W
    }
  if(N==K)
    W -= 0.5*Bp*Cpfj;
  W_Chomo[j] = W;                                     // Store total weight factor
  }
//                  Cycle for Heteronuclear E-COSY Phase Cycle
//   C. Griesinger, O.W. Sorensen and R.R. Ernst, JCP, 85, 6837-6852 (1986)
// Equations (47) and (48), both on page 6848

// phi = j * pi/N + del   * pi/(2*K) where j = [0, 2N-1]
//    j                N,K

//     K * sin[(K+1)*phi ] - (K+1) * sin(K*phi )
//                       j                   j           K
// W = ------------------------------------------ + del   * - sin(K*phi )
//  j                       2                      N,K  2         j
//             4 * sin [phi / 2]
//                          j

 double phi_hetero[P_cycl];                           // Pulse phase cycle angles
 double W_hetero[P_cycl];                             // Weighting factors
 double W_Chetero[P_cycl];                            // Weighting factors, comp.
 double SKfj, SKp1fj, Sfjo2;
 for(j=0; j<P_cycl; j++)
      {
      phij = double(j)*180.0/double(N);               // Calculate the angle phi
      if(N == K) phij += 180.0/double(2*K);
      phi_hetero[j] = phij;                           // Store phi
      phij *= PI/180.0;                               // Convert phi to radians
      SKfj = sin(double(K)*phij);                     // Start weight calculation
      SKp1fj = sin(double(K+1)*phij);
      Sfjo2 = sin(phij/2.0);
      W = double(K)*SKp1fj - double(K+1)*SKfj;
      W /= (4.0*Sfjo2*Sfjo2);
      if(K==N) W += double(K)*SKfj/2.0;
      W_hetero[j] = W;                                // Store the weighting factor
      }

 for(j=0; j<P_cycl; j++)
      {
      jpn = j+N;
      phijpn = double(jpn)*180.0/double(N);           // Calculate the angle phi
      if(N == K) phijpn += 180.0/double(2*K);
      phijpn *= PI/180.0;                             // Convert phi to radians
      SKfj = sin(double(K)*phijpn);                   // Start weight calculation
      SKp1fj = sin(double(K+1)*phijpn);
      Sfjo2 = sin(phijpn/2.0);
      W = double(K)*SKp1fj - double(K+1)*SKfj;
      W /= (4.0*Sfjo2*Sfjo2);
      if(K==N) W += double(K)*SKfj/2.0;
      W_Chetero[j] = W;                               // Store the weighting factor
      }
//                          Output the E-COSY Phase Cycle
 cout << “\n\t\t\t\tE-COSY Phase Cycle\n”;
 cout << “\t\t\t S.A Smith March 1993\n”;
 cout << “\t\t\t N = “ << N << “\t K = “ << K << “\n”;
 cout << “\n Step\t Homonuclear\t\t\t Heteronuclear”;
 cout << “\n\t\t phi\t W\t Wcomp”
      << “\t phi\t W\t Wcomp\n”;
 for(j=0; j<P_cycl; j++)
      {
      cout << “\n” << dec(j,10)
            << “ “ << dtoa(phi_homo[j], ‘f’, 10, 2)
            << “ “ << dtoa(W_homo[j], ‘f’, 10, 4)
            << “ “ << dtoa(W_Chomo[j], ‘f’, 10, 4)
            << “\t”
            << “ “ << dtoa(phi_hetero[j], ‘f’, 10, 2)
            << “ “ << dtoa(W_hetero[j], ‘f’, 10, 4)
            << “ “ << dtoa(W_Chetero[j], ‘f’, 10, 4);
      }
//                    Output these values into a FrameMaker Table
matrix mx(P_cycl, 7, 0.0);                            // Construct a matrix
 for(int i=0; i<P_cycl; i++)                          // Fill the matrix
      {
      mx.put(i,i,0);
      mx.put(phi_homo[i],i,1);
      mx.put(W_homo[i],i,2);
      mx.put(W_Chomo[i],i,3);
      mx.put(phi_hetero[i],i,4);
      mx.put(W_hetero[i],i,5);
      mx.put(W_Chetero[i],i,6);
      }
 FM_Mat_Tbl(“ecosy_pc.mif”, mx);                      // Output matrix to FM table
 cout << “\n\n”;                                      // Keep screen nice
 }
```

## 2.7.3 Discussion

**Get Coherence Order:** The program begins by asking the user which coherence order the E-COSY sequence is to be set for. The minimum allowed value is three. The number of phase cycle steps is determined and the values of N and K are set to the specified order (best sensitivity). The values of B0 and B1 are set to zero, they affect the homonuclear E-COSY cycle.

**Cycle for Homonuclear E-COSY:** In this section the homonuclear phase cycle and weighting factors are determined. The formulas implemented and the reference are in the program text. The first loop produces the phases and weights whereas the second loop determines the weights for the complimentary E-COSY experiment.

**Cycle for Heteronuclear E-COSY:** This code is the heteronuclear analog of the previous section.

**Output the E-COSY Phase Cycle:** The determined cycle is output to the console. The data is for the specified coherence order. Both the hetero- and homo-nuclear cases are output and weights for E-COSY and complementary E-COSY.

**Output to FrameMaker:** For convenience, the data is placed into a table in FrameMaker compatible format (MIF). First a matrix is filled with the data and then the function FM_Mat_Tbl puts the data into the file ecosy_pc.mif. The result of the program can be directly incorporated into any FrameMaker document, such as this one.

## 2.7.4 Results

**Table 3: E-COSY Phase Cycle N=K=3**

| STEP | $\phi$ | W | $W^{comp}$ | $\phi$ | W | $W^{comp}$ |
|---|---|---|---|---|---|---|
| | Homonuclear | | | HeteroNuclear | | |
| 0 | 0 | 2 | 0 | 30 | -3.73205 | .267949 |
| 1 | 60 | -1.5 | 0.5 | 90 | 0.5 | -0.5 |
| 2 | 120 | 0.5 | -1.5 | 150 | -.267949 | 3.73205 |
| 3 | 180 | 0 | 2 | 210 | .267949 | -3.73205 |
| 4 | 240 | 0.5 | -1.5 | 270 | -0.5 | 0.5 |
| 5 | 300 | -1.5 | 0.5 | 330 | 3.73205 | -.267949 |

*Table 19-1* E-COSY phase cycle suited for systems consisting predominately of 3 spin networks.

**Table 20: E-COSY Phase Cycle N=K=4**

| STEP | $\phi$ | W | $W^{comp}$ | $\phi$ | W | $W^{comp}$ |
|---|---|---|---|---|---|---|
| | Homonuclear | | | HeteroNuclear | | |
| 0 | 0 | 5 | 1 | 22.5 | -6.56854 | -.259892 |
| 1 | 45 | -3.41421 | -.585786 | 67.5 | .809957 | .361616 |
| 2 | 90 | 1 | 1 | 112.5 | -.361616 | -.809957 |
| 3 | 135 | -.585786 | -3.41421 | 157.5 | .259892 | 6.56854 |
| 4 | 180 | 1 | 5 | 202.5 | -.259892 | -6.56854 |
| 5 | 225 | -.585786 | -3.41421 | 247.5 | .361616 | .809957 |
| 6 | 270 | 1 | 1 | 292.5 | -.809957 | -.361616 |
| 7 | 315 | -3.41421 | -.585786 | 337.5 | 6.56854 | .259892 |

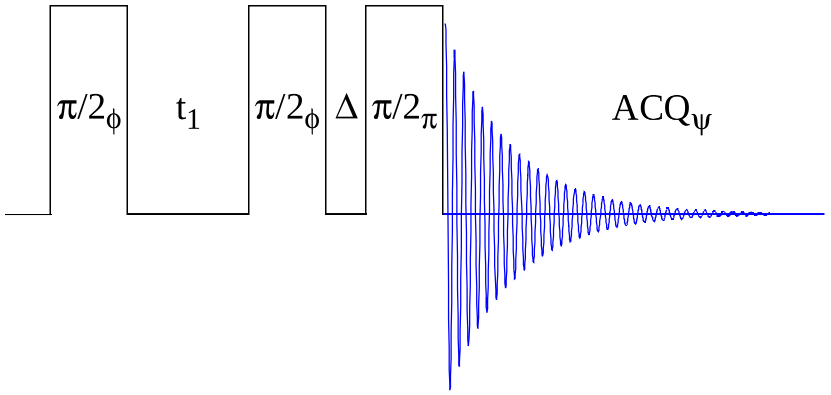
*Table 19-1* E-COSY phase cycle suited for systems consisting predominately of 4 spin networks, i.e. 4 mutually coupled spins.

## 2.8 E-COSY with Relaxation

### 2.8.1 Description

We now demonstrate how one may add in the effects of relaxation when simulating the E- COSY experiment. The pulse sequence and phase cycle will be the one used in our original example.

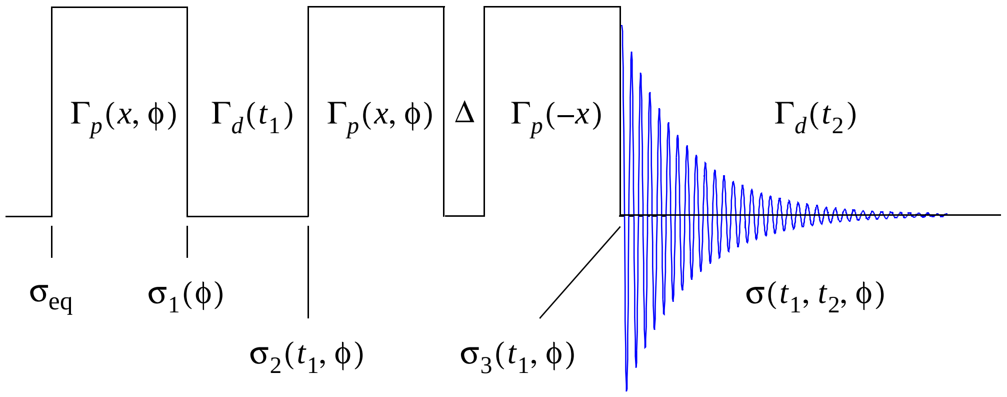
***E-COSY Pulse Sequence and Phase Cycle***



| W | $\phi$ | $\psi$ |
|---|---|---|
| 4 | 0 | 0 |
| 1 | 120 | 0 |
| 1 | 240 | 0 |
| 3 | 60 | 180 |
| 3 | 300 | 180 |

In the figure, W is a weighting factor, $\phi$ the phase angle of the first two pulses, and $\psi$ the phase angle of the detector. The evolution of the density matrix for the E-COSY sequence is depicted below, in this instance we shall keep our formalism entirely in Liouville space.

***E-COSY Pulse Sequence in Terms of Liouville Space Propagators***



The formulae for evolution of the density matrix in accordance with the previous figure are as follows.

$$|\sigma_1(\phi)\rangle = \Gamma_p(x, \phi)\sigma_{eq} = \Gamma_z(\phi)\Gamma_p(x)\Gamma_z^{-1}(\phi)\sigma_{eq} \tag{EQ 9}$$

$$|\sigma_2(t_1, \phi)\rangle = \Gamma_d(t_1)|\sigma_1(\phi)\rangle \tag{EQ 10}$$

$$|\sigma_3(t_1,\phi)\rangle = \Gamma_p(-x)\Gamma_p(x,\phi)|\sigma_2(t_1,\phi)\rangle$$
$$= \Gamma_p(-x)\Gamma_z(\phi)\Gamma_p(x)\Gamma_z^{-1}(\phi)|\sigma_2(t_1,\phi)\rangle \qquad \text{(EQ 11)}$$

$$|\sigma(t_1,t_2,\phi)\rangle = \Gamma_d(t_2)|\sigma_3(t_1,\phi)\rangle \qquad \text{(EQ 12)}$$

The FID is then computed by performing trace operations with a detection operator, $F_-$, which is also phase cycled.

$$FID(t_1,t_2,\phi,\psi) = \langle F_-^{\dagger}(\psi)|\sigma(t_1,t_2,\phi)\rangle \qquad \text{(12-1)}$$

The last step is that we sum the FID's over the phase cycle

$$FID(t_1,t_2) = \sum_{\phi,\psi}^{\text{cycle}} FID(t_1,t_2,\phi,\psi) \qquad \text{(12-2)}$$

With GAMMA, we could proceed implementing these equations as outlined and directly perform an E-COSY simulation. Obviously the task would then require some scheme for performing the phase cycle, perhaps involving a complicated algorithm and repetitive computations. Rather than using these first six equations as written, we now shuffle around the formulae and attempt to isolate components of the phase cycle in order to simplify implementation of the E-COSY sequence. From the first two equations we have

$$|\sigma_2(t_1,\phi)\rangle = \Gamma_d(t_1)\Gamma_z(\phi)\Gamma_p(x)\Gamma_z^{-1}(\phi)|\sigma_{eq}\rangle \quad .$$

Rotations about the z-axis cannot affect the equilibrium density matrix so the initial rotation super-operator may be safely removed. Furthermore, rotations about z commute with the time evolution propagators due to $[H_0,F_z] = 0$, so we may switch the order of these operations. The result is

$$|\sigma_2(t_1,\phi)\rangle = \Gamma_z(\phi)\Gamma_d(t_1)\Gamma_p(x)\Gamma_z^{-1}(\phi)|\sigma_{eq}\rangle \quad .$$

We define a new density matrix operators which are independent of the phase angle $\phi$,

$$|\sigma_1\rangle = \Gamma_p(x)|\sigma_{eq}\rangle \qquad \text{(12-3)}$$

and

$$|\sigma_2(t_1)\rangle = \Gamma_d(t_1)|\sigma_1\rangle \quad , \qquad \text{(12-4)}$$

so that the previous formula, (EQ 2), becomes simply

$$|\sigma_2(t_1,\phi)\rangle = \Gamma_z(\phi)|\sigma_2(t_1)\rangle \quad . \qquad \text{(12-5)}$$

A cancellation of the rotation superoperator occurs when the next step is formulated. From (EQ 3),

$$|\sigma_3(t_1,\phi)\rangle = \Gamma_p(-x)\Gamma_z(\phi)\Gamma_p(x)\Gamma_z^{-1}(\phi)|\sigma_2(t_1,\phi)\rangle$$

$$\sigma_3(t_1,\phi) = \Gamma(-x)\Gamma_z(\phi)\Gamma_p(x)|\sigma_2(t_1)\rangle \tag{12-6}$$

Note that the phase cycle is explicitly removed from the t1 incrementation at this point. We now shift our attention to the last sequence step and the detection, propagation for the $t_2$ delay followed by multiplication with a detection operator respectively. We have for detection

$$F_{-}(\psi)\sigma(t_1,t_2,\phi) = R_z(\psi)F_{-}R_z^{-1}(\psi)\sigma(t_1,t_2,\phi) \quad . \tag{12-7}$$

where we can utilize the relationship $Tr\{AB\} = Tr\{BA\}$ ,

$$\begin{aligned}\langle F_{-}^{\dagger}(\psi)|\sigma(t_1,t_2,\phi)\rangle &= Tr\{R_z(\psi)F_{-}R_z^{-1}(\psi)\sigma(t_1,t_2,\phi)\}\\ &= Tr\{F_{-}R_z^{-1}(\psi)\sigma(t_1,t_2,\phi)R_z(\psi)\}\end{aligned}$$

and include the $t_2$ delay step.

$$\begin{aligned}\langle F_{-}^{\dagger}(\psi)|\sigma(t_1,t_2,\phi)\rangle &= \langle F_{-}^{\dagger}|\Gamma_z^{-1}(\psi)\Gamma_d(t_2)|\sigma_3(t_1,\phi)\rangle\\ &= \langle F_{-}^{\dagger}|\Gamma_d(t_2)\Gamma_z^{-1}(\psi)|\sigma_3(t_1,\phi)\rangle\end{aligned} \quad . \tag{12-8}$$

Thus, the phase cycle over $\psi$ has been isolated from the detection and $t_2$ incrementation. If we again focus on the components which depend upon the phase cycle, we have

$$\Gamma_z^{-1}(\psi)|\sigma_3(t_1,\phi)\rangle = \Gamma_z^{-1}(\psi)\Gamma_p(-x)\Gamma_z(\phi)\Gamma_p(x)|\sigma_2(t_1)\rangle \quad . \tag{12-9}$$

The entire E-COSY pulse sequence phase cycle is contained in the above equation and we can formulate a new propagator which contains it. Letting

$$\Gamma_{mix}(\phi,\psi) = \Gamma_z^{-1}(\psi)\Gamma_p(-x)\Gamma_z(\phi)\Gamma_p(x) \quad , \tag{12-10}$$

The trace equation becomes,

$$\langle F_{-}^{\dagger}(\psi)|\sigma(t_1,t_2,\phi)\rangle = \langle F_{-}^{\dagger}|\Gamma_d(t_2)\Gamma_{mix}(\phi,\psi)\sigma_2(t_1)\rangle \quad . \tag{12-11}$$

In order to compute our 2-dimensional spectrum we need to sum FID's over each phase cycle according to (0-6)

$$FID(t_1,t_2) = \sum_{\phi,\psi}^{\text{cycle}} FID(t_1,t_2,\phi,\psi)$$

## 2.8.2 Program

```
/* ecosy5.cc ******************************************************************-*-c++-*-
**                                                                                    **
**                    Example Program for the GAMMA Library                          **
**                                                                                    **
** This program simulates an E-COSY experiment with complete                         **
** phase cycling. The pulses are taken to be ideal and the                           **
** effects of relaxation are included during t1 and t2.                              **
**                                                                                    **
** See: C. Griesinger, O.W. Sorensen, and R.R. Ernst, JMR,                           **
** 75, 474-492, (1987), "Practical Aspects of the E.COSY                             **
** Technique. Measurement of Scalar Spin-Spin Coupling                               **
** Constants in Peptides". The required phase cycle is                               **
** found on page 477 of this reference.                                              **
**                                                                                    **
** Note: This program treats only homonuclear spin systems.                          **
**      Superoperators are used to implement the phase cycle                         **
**      which speeds up the computation for small systems                            **
**      but this program is cumbersome for large systems.                            **
**                                                                                    **
************************************************************************************** */

#include "gamma.h"
//                          Define Constants
 const P_cycl = 12;                                  // Phase cycle length
 const double P_mix[12] =                            // Pulse phase cycle

{0,60,0,60,0,60,0,300,120,300,240,300};
//                          Begin Program
main (int argc, char* argv[])

 {
 cout << "\n\tSimulation of an E-COSY spectrum in TPPI mode\n";
 cout << "\t (Hard Pulses and Relaxation During Delays)\n";
//                     Read in Spectral Parameters
sys_dynamic sys;                                     // Dynamic spin system
 query_system(argc, argv, 1, sys);                   // Read system from file
 int t1pts, t2pts;
 query_parameter(argc, argv, 2,                      // Get number FID of points
        "\n\tAcquisition Size? ", t2pts);
 t1pts = 2*t2pts;                                    // Set t1 size for TPPI
 String Isotype = sys.symbol(0);                     // Get isotope type of 1st spin
 int isoset = query_isotope(sys, Isotype);           // Choose an isotope type
 double offset=query_offset(sys,isoset,1);           // Ask for an offset frequency
//                    Set Up the Isotropic Hamiltonian
String J;
 query_parameter(argc, argv, 3,                      // Weak or strong coupling
        "\n\tWeak or strong coupling (w/s)? ", J);
 gen_op H;                                           // Set Hamilitonian for
 if (J == "w") H = Hcs(sys) + HJw(sys);              / strong or weak coupling
else       H = Hcs(sys) + HJ(sys);
//           Set Up the Relaxation Superoperator, Liouvillian
 super_op R;
 query_parameter(argc, argv, 4,                      // Dipolar relaxation
 "\n\tInclude Dipolar Relaxation (y/n)? ", J);
 if(J == "y") R += RDD(sys, H);
 query_parameter(argc, argv, 5,                      // CSA relaxation
 "\n\tInclude CSA Relaxation (y/n)? ", J);
 if(J == "y") R += RCC(sys, H);
 query_parameter(argc, argv, 6,                      // Dipole-CSA relaxation
 "\n\tInclude Dipole-CSA Relaxation (y/n)? ", J);
 if(J == "y") R += RDCX(sys, H);
 complex icmplx(0,1);                                // z = 0 + 1i
 super_op L = icmplx*Hsuper(H);                      // L = -i*[Ho, ] (rad/sec)
 L += R;                                             // Full Liouvillian
 double lwhh = LWhh_DD_max(sys);                     // Expected dipolar linewidth
 double NyqF =                                       // Choose a Nyquist frequency
        query_Nyquist(sys,isoset,lwhh);
 double t2dt = 1.0/(2.0*NyqF);                       // Dwell time, quadrature
 double t1dt = t2dt/2.0;                             // t1 time increment, TPPI
//          Set Up Other Necessary Operators, Superoperators
 super_op eLt1 = exp(L, -t1dt);                      // Delay t1 relaxation superop
 gen_op sigma0 = sigma_eq(sys);                      // Density matrix at equilib
 set_trace(sigma0, 1.0);
 super_op G_d1 = R_prop(eLt1, sigma0);               // Delay t1 relaxation prop
 gen_op RTPPI = Rz(sys,90);                          // Rotation for TPPI phasing
 RTPPI.Op_base(H);                                   // Put op in Ho eigenbasis
 G_d1 = U_transform(RTPPI)*G_d1;                     // Delay t1 + TPPI propagator
 gen_op D = Fm(sys);                                 // Detector to F-
 gen_op Upx = Ixypuls_U(sys, 0, +90);                // Propagator for x pulse
 gen_op Upmx = Ixypuls_U(sys, 0, -90);               // Propagator for -x pulse
 acquire ACQ(D, L, t2dt);                            // Prepare for acquisitions
 gen_op U_mix;                                       // Temporary mixing propagator
 super_op G_mix;                                     // Phase cycle superoperator
 gen_op sigma3;
//            Construct Mixing, Phase Cycle Superoperator
 double conv = acos(-1)/180;
 double P_det = 0;
 cout << "\n\tConstructing Phase Cycle Superoperator\n";
 cout << "\n\t# scan                               betareference\n";
 for ( int i=0; i<P_cycl; i++ )
      {
      cout <<"\t" << i <<"                         "<<P_mix[i]<<""<<P_det<<"\n";
      U_mix = Rz(sys,-P_det);                        // Detector phase cycle
```

```
     U_mix *= Upmx;                                  // 3rd 90 Pulse (-x)
     U_mix *= Rz(sys,+P_mix[i]);                     // Phase shift pulses 1 & 2
     U_mix *= Upx;                                   // 2nd 90 Pulse
     U_mix.Op_base(H);                               // Put in eigenbasis of Ho
     G_mix += U_transform(U_mix);                    // Add to U transform supero
     P_det = acos(-cos(P_det*conv))/conv;            // Adjust detector phase
     }
//                         Apply Pulse Sequence
 File ecosy;                                         // Declare and open file
 ecosy.open("ecosy.dat",
 io_writeonly, a_create);
 block_1D t2BLK(t2pts);                              // Set 1D block for output
 gen_op sigma1 = evolve(sigma0, Upx);                // Apply first (PI/2)x pulse
 gen_op sigma2 = sigma1;                             // Initial sigma2 (t1 = 0)
 for (int t1=0; t1<t1pts; t1++)                      // Loop over all t1 increments
     {
     sigma3 = evolve(sigma2, G_mix);                 // Superop phase cycle
     ACQ(sigma3,t2BLK);                              // Collect the FID
     Felix(ecosy, t2BLK);                            // Output block: Felix
     evolve_ip(sigma2, G_d1);                        // Evolution next t1
     }
 ecosy.close();                                      // Close file
 }
```

### 2.8.3 Discussion

**Define Constants:** In this program the E-COSY phase cycle are explicitly defined; A 12 step cycle and associated phases in accordance with Figure 19-4.

**Read in Parameters:** As in prior simulations the spin system is read in from a disk file, the name may be supplied directly when running the program. The user is also asked for the acquisition size, whether to use a strong or weak coupling Hamiltonian, any desired offset, and a Nyquist frequency. Note that because TPPI will be used for phase sensitivity along the t1/f1 axis, twice as many points are taken along this axis that along the t2 axis, and the t1 incrementation time is half of the dwell time.

**Set up Operators, Superoperator:** The Hamiltonian is first formulated for either strong or weak coupling depending upon the input choice. Next, the $t_1$ time increment propagator is formulated. Here, in order to produce an acquisition with TPPI, a 90 degree rotation about the z-axis is added so that each incremented step includes the TPPI phase shift. The detector is set to F- in the next line. Propagators for the two 90 pulses are now computed, the first along x and the second along -x. A temporary operator is declared for used in summing over the phase cycle. The next line declares a superoperator which will ultimately account for the phase cycle. Finally, a working density matrix is specified.

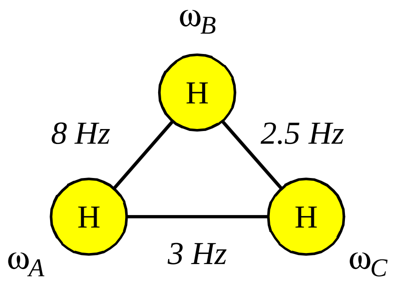
**Construct Mixing, Phase Cycle Superoperator:** As outlined mathematically at the start of this section, the entire phase cycle can be removed from the t1 and t2 looping by use of a superoperator. In this section, the phase cycle loop is applied and the appropriate superoperator constructed. The loop goes over the twelve steps of the cycle (defined in the constants section) and at each step the phases are output to the terminal. Initially the detector phase is set to zero. The propagators for each step are then multiplied together in the opposite order of the pulse sequence due to the ordering of the unary *= step. At the end of each cycle, the unitary transform superoperator is summed to account for the mixing propagator. The last step in this section adjusts the detector phase to +/- 180.

**Apply Pulse Sequence:** Since the superoperator intrinsically contains the phase cycling, the pulse sequence application is quite simple. A file name ecosy is created and opened. A 1D data block called t2BLK is constructed to temporarily store the FID. Following the pulse sequence diagram, the initial density matrix is set to equilibrium. The next step is to apply a 90 degree pulse (without phase cycling) along the x-axis. This is evolved next by the propagator which accounts for $t_1$ incrementation as well as TPPI phase adjustments, but it is not done on the first step through the $t_1$ increments, at $t_1$=0. Rather, this step is done at the end of the loop. The next step is evolution under the superoperator which contains all the phase cycling. This density matrix, sigma3, is then used to simulate an FID which is subsequently sent to the file ecosy.dat in Felix format. Again, the last step of the loop is actually the earlier t1 incrementation. The last program steps close the file, get the spectrometer frequency, output Felix parameters necessary for spectral workup, and clear the screen.

## 2.8.4 Spin System

For this example we will first treat a the simple 3 spin proton system which was investigated in a previous example.

***3-Spin Proton System for Homonuclear E-COSY with Relaxaiton***



**ecosy5.dsys**

```
SysName (2) : ECOSY1A- Name of the Spin System
NSpins  (0) : 3             - Number of Spins in the System
Iso(0)  (2) : 1H            - Spin Isotope Type
Iso(1)  (2) : 1H            - Spin Isotope Type
Iso(2)  (2) : 1H            - Spin Isotope Type
PPM(0)  (1) : -.1           - Chemical Shifts in PPM
PPM(1)  (1) : 0.0           - Chemical Shifts in PPM
PPM(2)  (1) :  .1           - Chemical Shifts in PPM
J(0,1)  (1) : 8.0           - Coupling Constants in Hz
J(0,2)  (1) : 3.0           - Coupling Constants in Hz
J(1,2)  (1) : 2.5           - Coupling Constants in Hz
Omega   (1) : 500           - 1H Spectrometer Frequency in MHz
```

## 2.8.5 Workup

The 2D-data sets produced from this simulation were processed with the program Felix. Each data set was subjected to the following Felix macro, called ecosy1.mac (the !comments to the right are not to be included in the macro - these are no longer allowed in the current Felix version.). Note that this macro ***assumes that the simulation had an acquisition size of 512 points.***

```
def datfile ecosy                   ! simulated data file name “ecosy.dat”
def matfile ecosy                   ! matrix file name “ecosy.mat”
def t1max 1024                      ! dimension t1 size
def t2max 512                       ! dimension t2 size
def total 1024
ty Opening source file “ &datfile “ .....! comment showing which data file
ty Opening matrix file “ &matfile “ .....! comment showing which matrix file
cmx                                 ! close any existing matrix
cl                                  ! close any existing dat files
mat &matfile write                  ! open matrix ecosy.mat
ty Filling matrix with data ...     ! comment that data processing starting
lb 4                                ! set line broadening for apodization
ph0 90                              ! set zero order phase correction
ph1 0                               ! set first order phase correction
ty Complex-FT of t2 into F2 ...     ! comment starting F2 transformations
for row 1 &t1max                    ! loop through all t1 rows
re &datfile                         ! read block of ecosy.dat file
si &t2max                           ! set the size
em                                  ! exponential multiplication
sb &t2max 90                        ! sine bell windowing function
zf &total                           ! zero fill to total size
ft                                  ! complex FFT
red                                 ! reduce data to real
sto 0 &row                          ! store processed data into matrix
next                                ! loop back for next block
ty Real-FT of t1 into F1 ......     ! comment for t1 workup
for col 1 &total                    ! loop through all columns
loa &col 0                          ! retrieve column from matrix
si &t1max                           ! specify the size
em                                  ! exponential multiplication
sb &t1max 90                        ! sine bell
zf 2048                             ! zero fill (will loose with real transform)
rft                                 ! real FFT
ph                                  ! apply phase adjustment
red                                 ! reduce data to real
sto &col 0                          ! store back into matrix
next                                ! get next column
ty Plotting contours ..........     ! comments for processing point
ty zooming cross peaks ........
cmx                                 ! clear matrices
mat &matfile read                   ! open ecosy.mat for reading only
lim 1 1 1024! set plot limits
lim 2 1 1024
lvl 5e-4                            ! set scaling
cpn 1                               ! contour positive and negative peaks
pen 1                               ! set first pen to pen 1
nl 4                                ! four contour levels
cyc 2                               ! pen cycle
```

```
cp                                  ! contour plot
ty Processing finished ........     ! comment for processing end
end
```

Prior to running this Felix macro the matrix ecosy.mat must be constructed with the following Felix command

```
bld ecosy 2 1024 1024 0             ! real array
```

which construct a 2 dimensional array (1K x 1K) of real numbers. After processing the matrix axes are set with the rmx command (used twice, once for each axis).

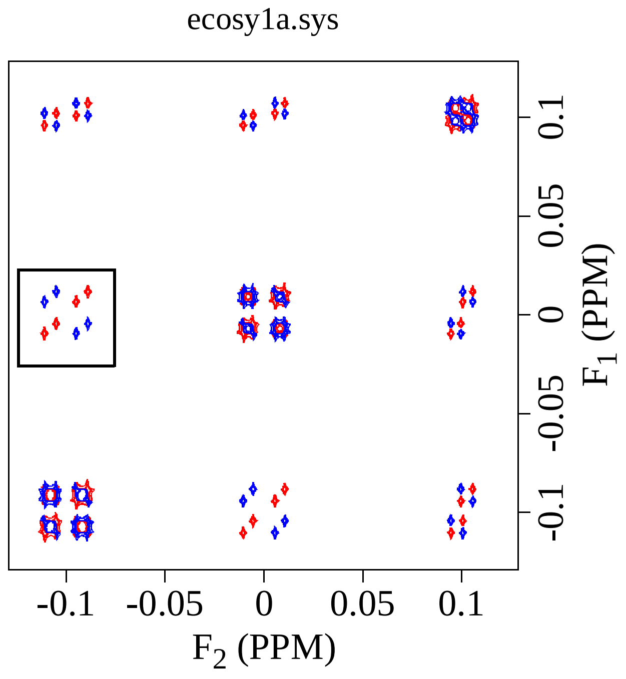
```
rmx
   1(2)                             ! specify the dimension (1 or 2)
   500! spectrometer frequency
   128                              ! spectral width
   3                                ! PPM plotted on axis
   512                              ! reference point
   0                                ! reference point value
   F2(F1)                           ! axis label (F2 or F1)
```

The simulated E-COSY spectra were placed into this FrameMaker document in encapsulated Postscript format which resulted from sending the HPGL output of Felix through the filter program hpgltoeps provided by FrameMaker. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
hdv felix.hpgl                      ! hardcopy device is file felix.hpgl
hpm 32                              ! hardcopy plot mode is hpgl
hcp                                 ! produce the plot
```

## 2.8.6 Results

***Simulated E-COSY Spectra on a 3-Spin Proton System***



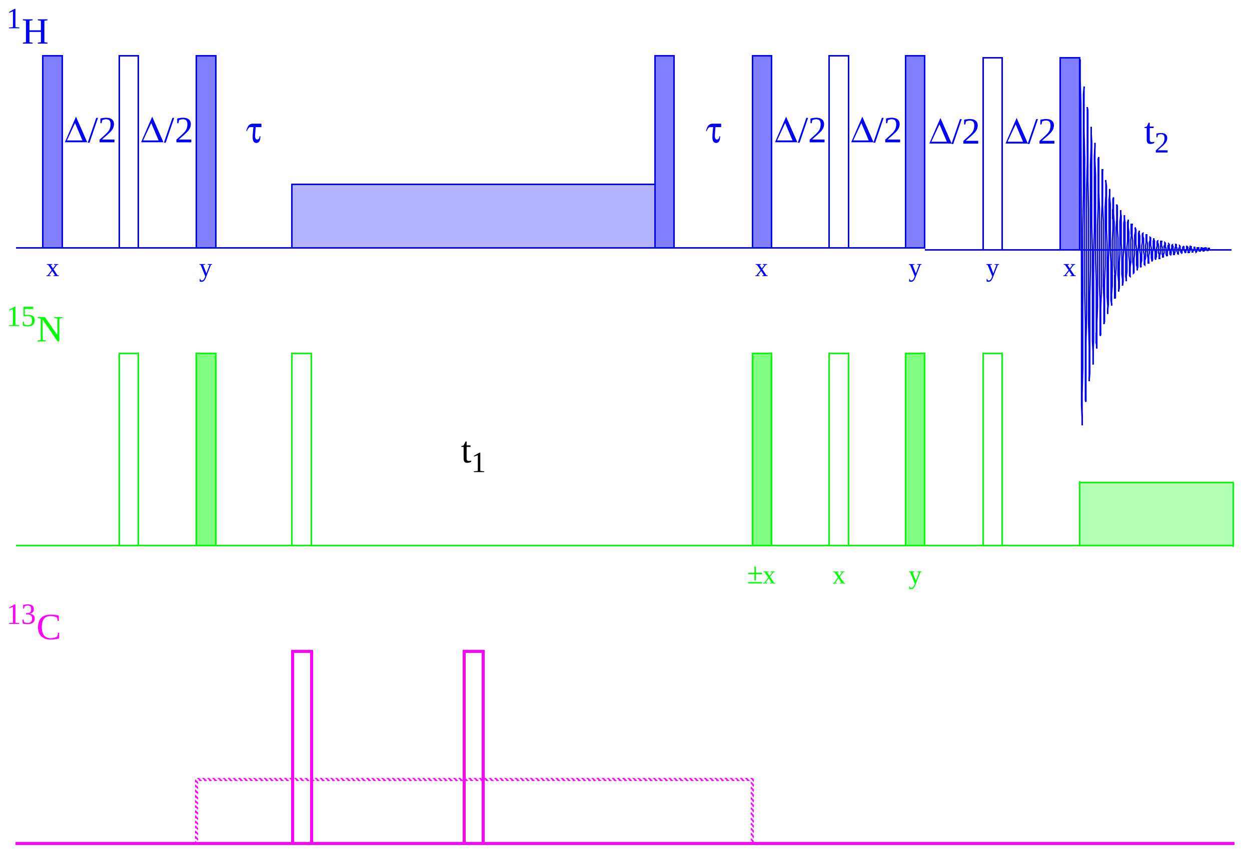
The simulation was repeated for the two 3-spin systems, the input spin systems files are used to label the plots. Each plot was output from Felix, placed into FrameMaker MIF format, rescaled and annotated, and finally placed onto this page. All were performed with weak coupling.

## 2.9 E-COSY with Phase Cycling

### 2.9.1 Description

We now move up to a more sophisticated simulation, that of an E- COSY (Exclusive COrrelation SpectroscopY) experiment[1]. The pulse sequence and phase cycle are given below.

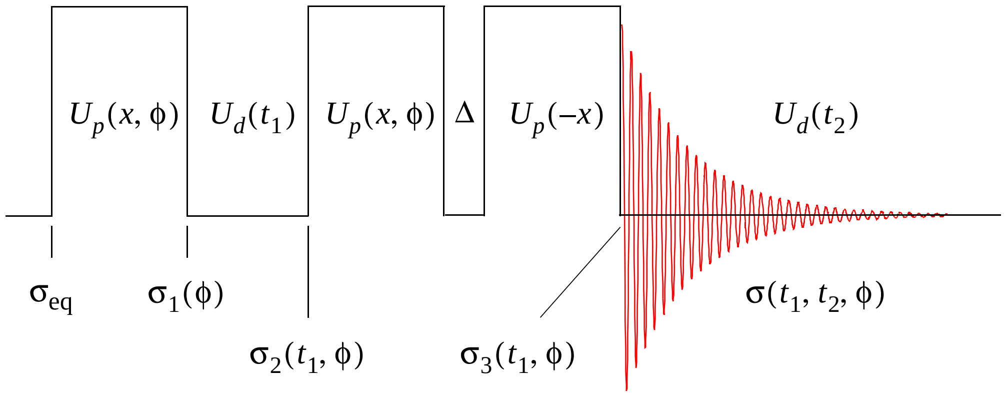
Sorensen Heteronuclear E-COSY Pulse Sequence



1. C. Griesinger, O.W. Sørensen, and R.R. Ernst (1985), *JACS*, **107**, 6394-6396, “Two-Dimensional Correlation of Connected NMR Transitions”.
C. Griesinger, O.W. Sørensen, and R.R. Ernst (1986), *J. Chem. Phys.*, **85**, 6837-6851, “Correlation of connected transitions by two-dimensional NMR spectroscopy”.
C. Griesinger, O.W. Sørensen, and R.R. Ernst (1987), *JMR*, **75**, 474-492, “Practical Aspects of the E.COSY Technique. Measurement of Scalar Spin-Spin Coupling Constants in Peptides”.

In the figure, W is a weighting factor, $\phi$ the phase angle of the first two pulses, and $\psi$ the phase angle of the detector. As we must employ a phase cycling scheme, we first consider the necessary mathematics for performing such a simulation, ultimately employing superoperators in order to simplify the mathematical formalism and reduce computation time. The evolution of the density matrix for the E-COSY sequence in terms of propagators is depicted below.

E-COSY Sequence in Terms of Hilbert Space Propagators



Although the previous figure does not show the detector phase angle, $\psi$, it will become explicit when formulating the acquisition. The formulae for evolution of the density matrix in accordance with the previous figure are as follows.

$$\sigma_1(\phi) = U_p(x, \phi)\sigma_{eq}U_p^{-1}(x, \phi) = R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_{eq}R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi) \quad \text{(EQ 13)}$$

$$\sigma_2(t_1, \phi) = U_d(t_1)\sigma_1(\phi)U_d^{-1}(t_1) \quad \text{(EQ 14)}$$

$$\begin{aligned}\sigma_3(t_1, \phi) &= U_p(-x)U_p(x, \phi)\sigma_2(t_1, \phi)U_p^{-1}(x, \phi)U_p^{-1}(-x)\\ &= U_p(-x)R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_2(t_1, \phi)R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x)\end{aligned} \quad \text{(EQ 15)}$$

$$\sigma(t_1, t_2, \phi) = U_d(t_2)\sigma_3(t_1, \phi)U_d^{-1}(t_2) \quad \text{(EQ 16)}$$

The FID is then computed by performing trace operations with a detection operator, $F_-$, which is also phase cycled.

$$FID(t_1, t_2, \phi, \psi) = Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} \quad \text{(EQ 17)}$$

The last step is that we sum the FID's over the phase cycle

$$FID(t_1, t_2) = \sum_{\phi,\psi}^{\text{cycle}} FID(t_1, t_2, \phi, \psi) \quad \text{(EQ 18)}$$

With GAMMA, we could proceed implementing these equations as outlined and directly perform an E-COSY simulation. Obviously the task would then require some scheme for performing the phase cycle, perhaps involving a complicated algorithm and repetitive computations. Rather than using these first six equations as written, we now shuffle around the formulae and attempt to isolate components of the phase cycle in order to simplify implementation of the E-COSY sequence. From the first two equations we have

$$\sigma_2(t_1, \phi) = U_d(t_1)R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_{eq}R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi)U_d^{-1}(t_1) \quad .$$

Rotations about the z-axis cannot affect the equilibrium density matrix so the inner rotation operators may be safely removed. Furthermore, rotations about z commute with the time evolution propagators due to $[H_0,F_z] = 0$, so we may switch the order of these operations. The result is

$$\sigma_2(t_1, \phi) = R_z(\phi)U_d(t_1)U_p(x)\sigma_{eq}U_p^{-1}(x)U_d^{-1}(t_1)R_z^{-1}(\phi) \quad .$$

We define a new density matrix operators which are independent of the phase angle $\phi$,

$$\sigma_1 = U_p(x)\sigma_{eq}U_p^{-1}(x) \qquad \text{(EQ 19)}$$

and

$$\sigma_2(t_1) = U_d(t_1)\sigma_1U_d^{-1}(t_1) \quad , \qquad \text{(EQ 20)}$$

so that the previous formula, (EQ 14), becomes simply

$$\sigma_2(t_1, \phi) = R_z(\phi)\sigma_2(t_1)R_z^{-1}(\phi) \quad . \qquad \text{(EQ 21)}$$

A cancellation of these rotation operators occurs when the next step is formulated. From (EQ 15),

$$\sigma_3(t_1, \phi) = U_p(-x)R_z(\phi)U_p(x)R_z^{-1}(\phi)\sigma_2(t_1, \phi)R_z(\phi)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x)$$

$$\sigma_3(t_1, \phi) = U_p(-x)R_z(\phi)U_p(x)\sigma_2(t_1)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x) \qquad \text{(EQ 22)}$$

Note that the phase cycle is explicitly removed from the t1 incrementation at this point. We now shift our attention to the last sequence step and the detection, propagation for the $t_2$ delay followed by multiplication with a detection operator respectively. We have for detection

$$F_-(\psi)\sigma(t_1, t_2, \phi) = R_z(\psi)F_-R_z^{-1}(\psi)\sigma(t_1, t_2, \phi) \quad . \qquad \text{(EQ 23)}$$

where we can utilize the relationship $Tr\{AB\} = Tr\{BA\}$ ,

$$Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} = Tr\{R_z(\psi)F_-R_z^{-1}(\psi)\sigma(t_1, t_2, \phi)\}$$

$$= Tr\{F_-R_z^{-1}(\psi)\sigma(t_1, t_2, \phi)R_z(\psi)\}$$

and include the $t_2$ delay step.

$$\begin{aligned} Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} &= Tr\{F_- R_z^{-1}(\psi)U_d(t_2)\sigma_3(t_1, \phi)U_d^{-1}(t_2)R_z(\psi)\} \\ &= Tr\{F_- U_d(t_2)R_z^{-1}(\psi)\sigma_3(t_1, \phi)R_z(\psi)U_d^{-1}(t_2)\} \end{aligned} \quad . \qquad \text{(EQ 24)}$$

Thus, the phase cycle over $\psi$ has been isolated from the detection and $t_2$ incrementation. If we again focus on the components which depend upon the phase cycle, we have

$$\begin{aligned} R_z^{-1}(\psi)\sigma_3(t_1, \phi)R_z(\psi) &= \\ &= R_z^{-1}(\psi)U_p(-x)R_z(\phi)U_p(x)\sigma_2(t_1)U_p^{-1}(x)R_z^{-1}(\phi)U_p^{-1}(-x)R_z(\psi) \end{aligned} \quad . \qquad \text{(EQ 25)}$$

The entire E-COSY pulse sequence phase cycle is contained in the above equation and we can formulate a new propagator which contains it. Letting

$$U_{mix}(\phi, \psi) = R_z^{-1}(\psi)U_p(-x)R_z(\phi)U_p(x) \quad , \qquad \text{(EQ 26)}$$

The trace equation becomes,

$$Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} = Tr\{F_- U_d(t_2)U_{mix}(\phi, \psi)\sigma_2(t_1)U_{mix}^{-1}(\phi, \psi)U_d^{-1}(t_2)\} \quad . \qquad \text{(EQ 27)}$$

In order to compute our 2-dimensional spectrum we need to sum FID's over each phase cycle according to (EQ 18)

$$FID(t_1, t_2) = \sum_{\phi, \psi}^{\text{cycle}} FID(t_1, t_2, \phi, \psi)$$

and we have a different mixing propagator for each combination of $\phi$ and $\psi$. We now employ the power of superoperators in order to simplify this cycle. We form the unitary transformation superoperator equivalent to the mixing propagator. The equation then becomes,

$$Tr\{F_-(\psi)\sigma(t_1, t_2, \phi)\} = Tr\left\{F_- U_d(t_2)[\hat{\Gamma}_{mix}(\phi, \psi)\sigma_2(t_1)]U_d^{-1}(t_2)\right\} \quad . \qquad \text{(EQ 28)}$$

Now when we sum over the phase cycle we have

$$FID(t_1, t_2) = \sum_{\phi,\psi} FID(t_1, t_2, \phi, \psi)$$

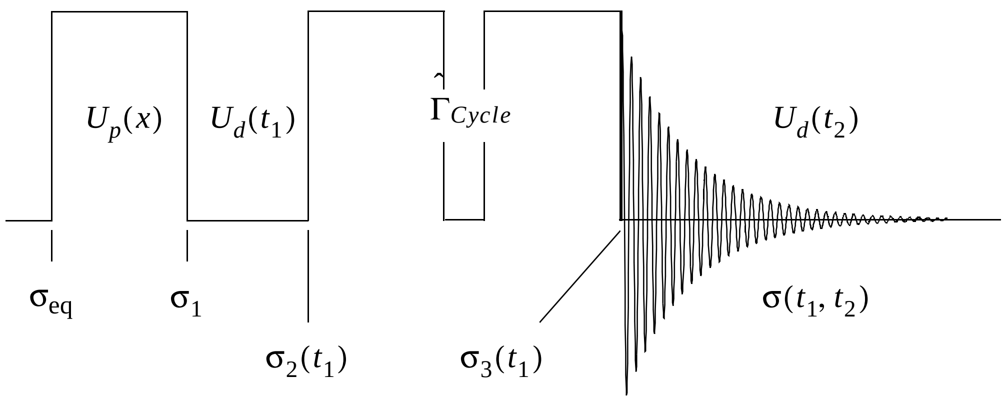
$$\sum_{\phi,\psi}^{cycles} Tr\left\{F_- U_d(t_2)\left\{\hat{\Gamma}(\phi,\psi)[U_d(t_1)U_p(x)\sigma_{eq}U_p^{-1}(x)U_d^{-1}(t_1)]\right\}U_d^{-1}(t_2)\right\}$$

$$Tr\left\{F_- U_d(t_2)\sum_{\phi,\psi}^{cycles}\hat{\Gamma}(\phi,\psi)[U_d(t_1)U_p(x)\sigma_{eq}U_p^{-1}(x)U_d^{-1}(t_1)]U_d^{-1}(t_2)\right\} \quad \text{(EQ 29)}$$

$$\left\{F_- U_d(t_2)(\hat{\Gamma}_{Cycle}[U_d(t_1)U_p(x)\sigma_{eq}U_p^{-1}(x)U_d^{-1}(t_1)]U_d^{-1}(t_2))\right\}$$

The phase cycle needed for the E-COSY simulation is now entirely contained in the superoperator. On a computational level, this means that the phase cycling loop can be removed from the loop over $t_1$ increments. The code becomes more concise and the computation more efficient[1]. The E-COSY pulse sequence in terms of the phase cycle superoperator can depicted as follows.

E-COSY Sequence Using Superoperator Phase Cycle



The simulation is performed exactly this way. We initially form the equilibrium density matrix, the pulse propagators, the $t_1$ propagator for $t_1$ incrementation time, and the superoperator representing the entire E-COSY phase cycle.

1. This is true only for small spin systems, <5 spins. As the system size increases it becomes more difficult for computers to handle superoperators due to the size of the Liouville space.

# 3 MQF-COSY

## 3.1 Introduction

The Multiple Quantum Filtered COSY experiment shares the same pulse sequence as the E-COSY sequence but employs different phase cycling. The pulse sequence is shown in the following figure.

MQF-COSY Pulse Sequence

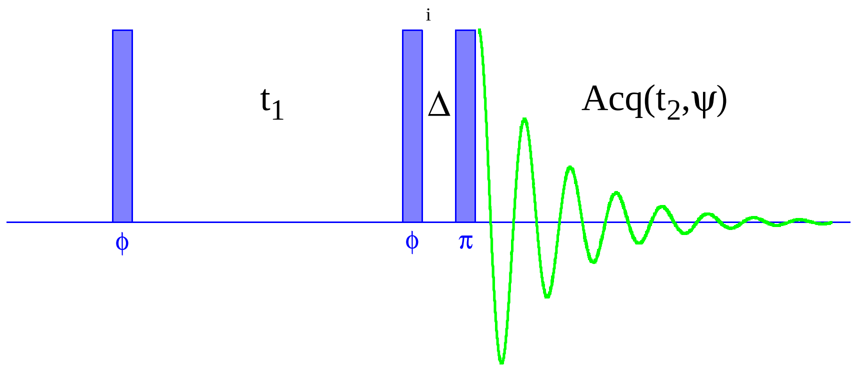


***Figure 0-27*** **The MQF-COSY pulse sequence. All three pulses are P/2, the phase cycle varies depending upon the coherence order selected.**

The required phase cycle depends upon the coherence order to be selected.The acquisition phase ψ typically oscillates between 0 and π, whereas pulse phases increment according to the formula

$$\Delta\phi = \frac{2\pi}{1p} \tag{29-1}$$

where $p$ is the coherence order. A few of these phase cycles are given below

MQF-COSY Phase Cycles

0QF

| φ | ψ |
|---|---|
| 0 | 0 |
| 90 | 0 |
| 180 | 0 |
| 270 | 0 |

1QF

| φ | ψ |
|---|---|
| 0 | 0 |
| 180 | 0 |

2QF

| φ | ψ |
|---|---|
| 0 | 0 |
| 90 | 180 |
| 180 | 0 |
| 270 | 180 |

3QF

| φ | ψ |
|---|---|
| 0 | 0 |
| 60 | 180 |
| 120 | 0 |
| 180 | 180 |
| 240 | 0 |
| 300 | 180 |

***Figure 0-28*** **MQF-COSY pulse sequence phase cycles for coherence orders 0-3.**

## 3.2 MQF-COSY with Superoperators & Felix Output

### 3.2.1 Description

This example will be a variation on a previous simulation of the E-COSY sequence but employs different phase cycling. The pulse sequence is given below.

MQF-COSY Pulse Sequence

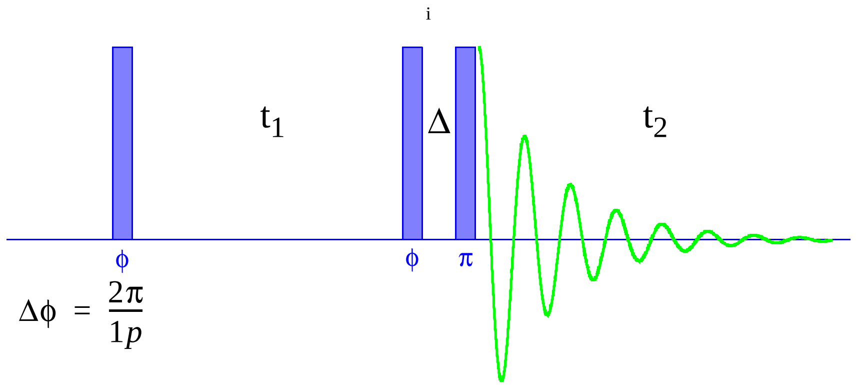


***Figure 0-29*** **The MQF-COSY pulse sequence. All three pulses are P/2, the phase cycle varies depending upon the coherence order selected.**

The phase cycle needed depends upon the coherence order to be selected. In generel, the acquisition phase $\psi$ is oscillating between 0 and $\pi$, whereas the pulse phases are incremented according to the formula

where $p$ is the coherence order. A few of these phase cycles are given below.

MQF-COSY Phase Cycles

0QF

| $\phi$ | $\psi$ |
|---|---|
| 0 | 0 |
| 90 | 0 |
| 180 | 0 |
| 270 | 0 |

1QF

| $\phi$ | $\psi$ |
|---|---|
| 0 | 0 |
| 180 | 0 |

2QF

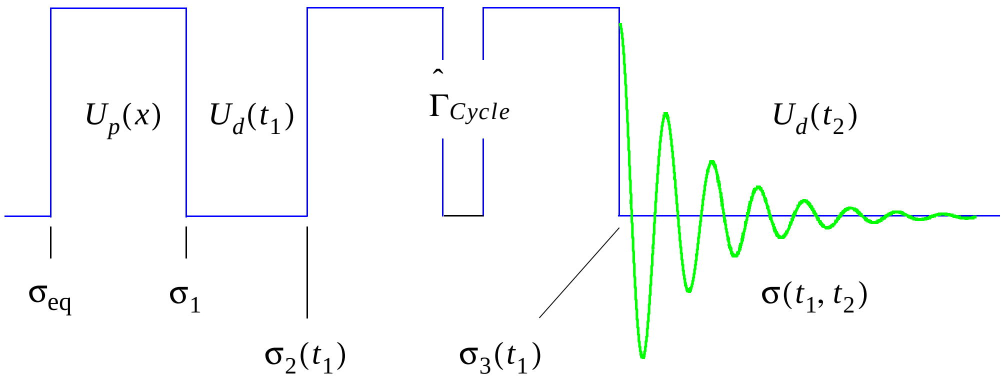
| $\phi$ | $\psi$ |
|---|---|
| 0 | 0 |
| 90 | 180 |
| 180 | 0 |
| 270 | 180 |

3QF

| $\phi$ | $\psi$ |
|---|---|
| 0 | 0 |
| 60 | 180 |
| 120 | 0 |
| 180 | 180 |
| 240 | 0 |
| 300 | 180 |

The MQF-COSY simulation will be performed in analagous fashiion to the previous E-COSY simulation using a superoperator for the phase cycling. The MQF-COSY pulse sequence in terms of the phase cycle superoperator can depicted as follows[1].

MQF-COSY Sequence Using Superoperator Phase Cycle



The simulation will be performed exactly this. We initially form the equilibrium density matrix, the pulse propagators, the $t_1$ propagator for $t_1$ incrementation time, and the superoperator representing the entire MQF-COSY phase cycle. The program by design will ask for the coherence order to be selected and generate the necessary phase cycle and its corresponding superoperator.

1. For the mathematical basis of this sequence utilizing superoperators see the E-COSY simulation and its description.

## 2.9.2 Program

```
/* mqfcosy.cc ***************************************************************-*-c++-*-
**                                                                                   **
**                    Example program for the GAMMA Library                          **
**                                                                                   **
** The Program reads a spin system (number of spins, chemical shifts                 **
** and J-coupling constants) from a file and simulates a Multiple Quantum            **
** Filtered COSY experiment complete with phase cycling.                             **
**                                                                                   **
***************************************************************************************/

#include “gamma.h”
//                              Define Constants

 const int t1pts = 256;                       // Number of t1 points
 const int t2pts = 128;                       // Number of t2 points
 const double t1dt = 0.002;                   // t1 time increment
 const double t2dt = 0.004;                   // t2 time increment
//                              Begin Program

main (int argc, char* argv[])
{
 cout << “\nSimulation of 2D-MQF-COSY spectra in TPPI mode\n”;
//                              Read in Parameters

String filename;                              // Name of spin system file
 query_parameter(argc, argv, 1,               // Get filename from command
          “\nSpin system filename? “, filename);  // line or ask for it
 spin_system sys;                             // Declare spin system sys
 sys.read(filename);                          // Read system from filename
 int MQF;
 query_parameter(argc, argv, 2,               // Get quantum order desired
             “\nQuantum order selected? “, MQF);
 MQF = abs(MQF);                              // Insure a non-negative value
 int P_cycl;                                  // Set up phase cycling
 double P_incr;
 if (MQF)
   {
   P_cycl = 2*MQF;                            // MQF > 0
   P_incr = 360/P_cycl;
   }
 else
   {
   P_cycl = 4;                                // Zero quantum filter
   P_incr = 90;                               // Supression up to 2QC
   }
 char J;
 query_parameter(argc, argv, 3,               // Weak or strong coupling
            “\nWeak or strong coupling (w/s)?”, J);
//                    Set Up Operators, Superoperator
gen_op H;                                     // Set Hamilitonian for
 if (J == ‘w’)                                // strong or weak coupling
 H = Hcs(sys) + HJw(sys);
 else
 H = Hcs(sys) + HJ(sys);
 gen_op Ud1 = Rz(sys,+90)*prop(H,t1dt);       // Delay t1 + TPPI propagator
 gen_op D = Fm(sys);                          // Detector to F-
 gen_op Upx = Ixypuls_U(sys, +90, 0);         // Propagator for x pulse
 gen_op Upmx = Ixypuls_U(sys, -90, 0);        // Propagator for -x pulse
 gen_op U_mix;                                // Temporary mixing propagator
 super_op G_mix;                              // Phase cycle superoperator
 gen_op sigma3;
//                Construct Mixing, Phase Cycle Superoperator
double conv = acos(-1)/180;
 double P_det = 0;
 double sign = -1;
 if(!MQF) sign = 1;                           // Constant phase ZQC
 double P_mix = 0;
 cout << “\n\tscan #\tbeta\treference\n”;
 for ( int i=0; i<P_cycl; i++ )
   {
   cout << “\t “ << i << “\t “<< P_mix <<“\t “<< P_det << “\n”;
   U_mix = Rz(sys,-P_det);                    // Detector phase cycle
   U_mix *= Upmx;                             // 3rd 90 Pulse (-x)
   U_mix *= Rz(sys,+P_mix);                   // Phase shift pulses 1 & 2
   U_mix *= Upx;                              // 2nd 90 Pulse
   U_mix.Op_base(H);                          // Put in eigenbasis of Ho
   G_mix += U_transform(U_mix);               // Add to U transform superop
   P_mix += P_incr;                           // Adjust pulse phase
   P_det = acos(sign*cos(P_det*conv))/conv;   // Adjust detector phase
   }
//                         Apply Pulse Sequence
File mqfcosy;                                 // Declare and open file
 mqfcosy.open(“mqfcosy.dat”,
 io_writeonly, a_create);
 block_1D t2BLK(t2pts);                       // Set 1D block for output
 gen_op sigma0 = sigma_eq(sys);               // Set density matrix equilibrium
 gen_op sigma1 = evolve(sigma0, Upx);         // Apply first (PI/2)x pulse
 gen_op sigma2 = sigma1;                      // Initial sigma2 (t1 = 0)
 for (int t1=0; t1<t1pts; t1++)               // Loop over all t1 increments
   {
   sigma3 = evolve(sigma2, G_mix);            // superop phase cycle
   FID(sigma3,D,H,t2dt,t2pts,t2BLK);          //acquisition
   Felix(mqfcosy, t2BLK);                     // output block: Felix
   evolve_ip(sigma2, Ud1);                    //evolution next t1
   }
 mqfcosy.close();                             // Close file
}
```

## 3.2.2 Discussion

**Define Constants:** In this program the dwell times and number of points to use on the $t_1$ and $t_2$ axes are defined as constants prior to entering the program. The size of the phase cycle (12) and the phase angles for the pulses are set as well here.

**Read in Parameters:** As in prior simulations the spin system is read in from a disk file, the name may be supplied directly when running the program. Also queried is whether to use a strong or weak coupling Hamiltonian.

**Read in Parameters:** As in prior simulations the spin system is read in from a disk file, the name may be supplied directly when running the program. Also queried is whether to use a strong or weak coupling Hamiltonian.

**Set up Operators, Superoperator:** The Hamiltonian is first formulated for either strong or weak coupling depending upon the input choice. Next, the $t_1$ time increment propagator is formulated. Here, in order to produce an acquisition with TPPI, a 90 degree rotation about the z-axis is added so that each incremented step includes the TPPI phase shift. The detector is set to F- in the next line. Propagators for the two 90 pulses are now computed, the first along x and the second along -x. A temporary operator is declared for used in summing over the phase cycle. The next line declares a superoperator which will ultimately account for the phase cycle. Finally, a working density matrix is specified.

**Construct Mixing, Phase Cycle Superoperator:** As outline mathematically at the start of this section, the entire phase cycle can be removed from the t1 and t2 looping by use of a superoperator. In this section, the phase cycle loop is applied and the appropriate superoperator constructed. The loop goes over the twelve steps of the cycle (defined in the constants section) and at each step the phases are output to the terminal. Initially the detector phase is set to zero. The propagators for each step are then multiplied together <u>in the opposite order of the pulse sequence</u> due to the ordering of the unary *= step. At the end of each cycle, the unitary transform superoperator is summed to account for the mixing propagator. The last step in this section adjusts the detector phase to +/- 180.

**Apply Pulse Sequence:** Since the superoperator intrinsically contains the phase cycling, the pulse sequence application is quite simple. A file name ecosy is created and opened. A 1D data block called t2BLK is constructed to temporarily store the FID. Following the pulse sequence diagram, the initial density matrix is set to equilibrium. The next step is to apply a 90 degree pulse (without phase cycling) along the x-axis. This is evolved next by the propagator which accounts for $t_1$ incrementation as well as TPPI phase adjustments, but it is not done on the first step through the $t_1$ increments, at $t_1$=0. Rather, this step is done at the end of the loop. The next step is evolution under the superoperator which contains all the phase cycling. This density matrix, sigma3, is then used to simulate an FID which is subsequently sent to the file ecosy.dat in Felix format. Again, the last step of the loop is actually the earlier t1 incrementation. The last program step simply closes the file.

### 3.2.3 Example Spin System

For the example we will utilize the a prototypical amino acid proton system pictured below.

***4-Spin "Amino Acid" Proton System for MQF-COSY Simulations***

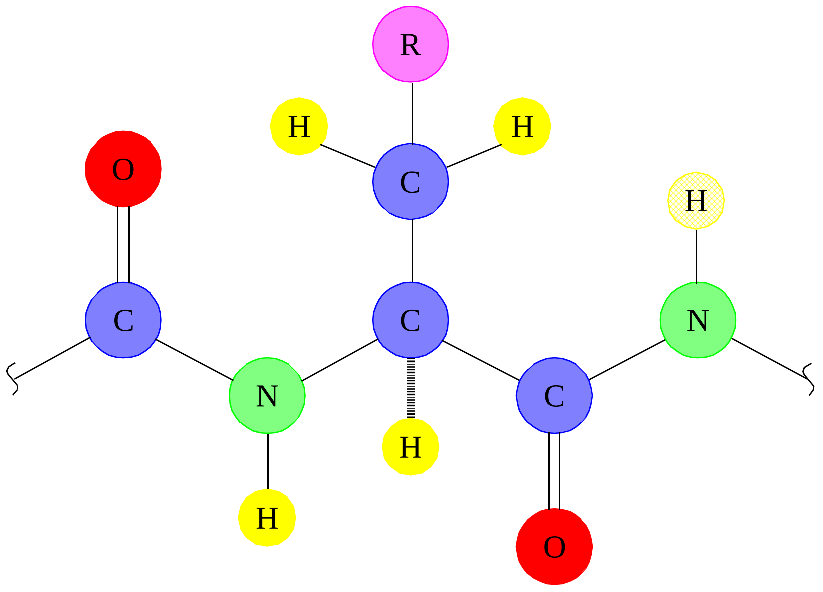


***Figure 0-30*** **The MQF-COSY simulation will be run on the above 4 spin system which contain appropriate proton shifts and coupling constants.**

The corresponding input spin system (ASCII) file read by the program is presented below.

**mqfcosy.sys**

```
SysName  (2) : system   - Name of the Spin System
NSpins   (0) : 4        - Number of Spins in the System
Iso(0)   (2) : 1H       - Spin Isotope Type
Iso(1)   (2) : 1H       - Spin Isotope Type
Iso(2)   (2) : 1H       - Spin Isotope Type
Iso(3)   (2) : 1H       - Spin Isotope Type
PPM(0)   (1) : -.420    - Chemical Shifts in PPM
PPM(1)   (1) : -.035    - Chemical Shifts in PPM
PPM(2)   (1) : .250     - Chemical Shifts in PPM
PPM(3)   (1) : .385     - Chemical Shifts in PPM
J(0,1)   (1) : 6.8      - Coupling Constants in Hz
J(0,2)   (1) : 0.0      - Coupling Constants in Hz
J(0,3)   (1) : -.9      - Coupling Constants in Hz
J(1,2)   (1) : 9.8      - Coupling Constants in Hz
J(1,3)   (1) : 0.0      - Coupling Constants in Hz
J(2,3)   (1) : -15      - Coupling Constants in Hz
Omega    (1) : 256      - Spectrometer Frequency in MHz (1H based)
```

## 3.2.4 Workup

The 2D-data sets produced from this simulation were processed with the program Felix. Each data set was subjected to the following Felix macro, called mqfcosy.mac (the !comments to the right are not to be included in the macro - these are no longer allowed in Felix.):

```
def datfile mqfcosy                    ! simulated data file name “ecosy.dat”
def matfile mqfcosy                    ! matrix file name “ecosy.mat”
def t1max 256                          ! dimension t1 size
def t2max 128                          ! dimension t2 size
def total 256
ty Opening source file “ &datfile “ ....! comment showing which data file
ty Opening matrix file “ &matfile “ ....! comment showing which matrix file
cmx                                    ! close any existing matrix
cl                                     ! close any existing dat files
mat &matfile write                     ! open matrix ecosy.mat
ty Filling matrix with data ...        ! comment that data processing starting
lb 2                                   ! set line broadening for apodization
ph0 90                                 ! set zero order phase correction
ph1 0                                  ! set first order phase correction
ty Complex-FT of t2 into F2 ...        ! comment starting F2 transformations
for row 1 &t1max                       ! loop through all t1 rows
re &datfile                            ! read block of ecosy.dat file
si &t2max                              ! set the size
em                                     ! exponential multiplication
sb &t2max 90                           ! sine bell windowing function
zf &total                              ! zero fill to total size
ft                                     ! complex FFT
red                                    ! reduce data to real
sto 0 &row                             ! store processed data into matrix
next                                   ! loop back for next block
ty Real-FT of t1 into F1 ......        ! comment for t1 workup
for col 1 &total                       ! loop through all columns
loa &col 0                             ! retrieve column from matrix
si &t1max                              ! specify the size
em                                     ! exponential multiplication
sb &t1max 90                           ! sine bell
zf 512                                 ! zero fill (will loose with real transform)
rft                                    ! real FFT
ph                                     ! apply phase adjustment
red                                    ! reduce data to real
sto &col 0                             ! store back into matrix
next                                   ! get next column
ty Plotting contours ..........        ! comments for processing point
ty zooming cross peaks ........
cmx                                    ! clear matrices
mat &matfile read                      ! open ecosy.mat for reading only
lim 1 1 256                            ! set plot limits
lim 2 1 256
lvl 1e-4                               ! set scaling
cpn 1                                  ! contour positive and negative peaks
pen 1                                  ! set first pen to pen 1
nl 4                                   ! four contour levels
cyc 2                                  ! pen cycle
cp                                     ! contour plot
ty Processing finished ........        ! comment for processing end
```

```
        end
```

Prior to running this Felix macro the matrix ecosy.mat must be constructed with the following Felix command

```
        bld ecosy 2 256 256 0                  ! real array
```

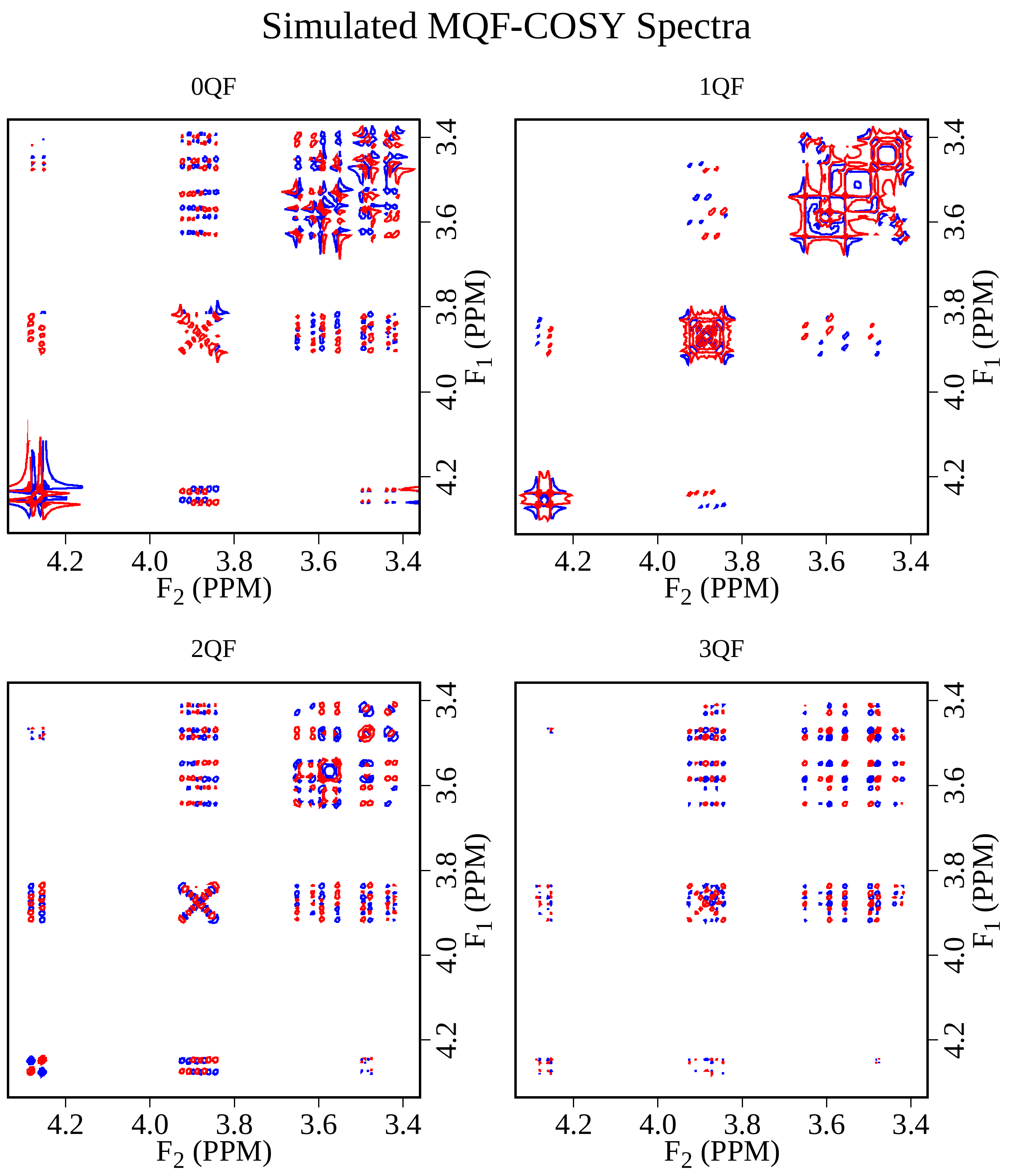
which construct a 2 dimensional array (256x256) of real numbers. After processing the matrix axes are set with the rmx command (used twice, once for each axis).

```
        rmx
           1(2)                                ! specify the dimension (1 or 2)
           256                                 ! spectrometer frequency
           200                                 ! spectral width
           3                                   ! PPM plotted on axis
           128                                 ! reference point
           0                                   ! reference point value
           F2(F1)                              ! axis label (F2 or F1)
```

The simulated E-COSY spectra were placed into this FrameMaker document in encapsulated Post-script format which resulted from sending the HPGL output of Felix through the filter program hpgltoeps provided by FrameMaker. To generate the hpgl contour plot file, the Felix commands are (once cp produces the screen plot)

```
        hdv felix.hpgl                         ! hardcopy device is file felix.hpgl
        hpm 32                                 ! hardcopy plot mode is hpgl
        hcp                                    ! produce the plot
```

## 3.2.5 Results


Simulated MQF-COSY Spectra
0QF
1QF
2QF
3QF
4.2
4.0
3.8
3.6
3.4
F2 (PPM)
F1 (PPM)

# 4 Mathematical Description

## 4.1 Overview

All functions provided in the NMR library are described mathematically as well as qualitatively. Since these are generally specific higher level functions we here provide the mathematical description which is at the core of these other descriptions.

## 4.2 Liouville Equation, Lab. Frame

Because we normally use the density matrix to describe the spin system we now look at its equation of motion.

## 4.3 Liouville Equation, Interaction Rep.

There are many instances in which the acting Hamiltonian consists of a time independent component as well as a time dependent component. In this case it is usually beneficial to switch into the interaction representation in order to simplify solution of the Liouville equation by effectively removing the time independent Hamiltonian component from the equation. Consider such a situation where the total Hamiltonian is given by

$$\boldsymbol{H}(t) = \boldsymbol{H}_I + \boldsymbol{H}_{II}(t) \tag{29-2}$$

so that the Liouville equation is then

$$ih\frac{d\sigma}{dt} = [\boldsymbol{H}(t), \sigma] = [\boldsymbol{H}_I + \boldsymbol{H}_{II}(t), \sigma] \quad . \tag{29-3}$$

Any operator can be switched into the interaction representation by

$$\hat{O} = e^{iH_I t} O e^{-iH_I t} \tag{29-4}$$

where operators in the interaction representation are designated with a hat, ^. The equation of motion is a direct analog of the original Liouville equation

$$ih\frac{d}{dt}\hat{\sigma} = [\hat{\boldsymbol{H}}_{II}(t), \hat{\sigma}] \tag{29-5}$$

as we will show momentarily. Note that from the switch into the interaction representation we obtain a new equation of motion which does not (explicitly) contain the original time independent component $\boldsymbol{H}_I$. There is no garuntee, but hopefully equation (29-5) is easier to solve than (29-3) and once accomplished the density matrix in the interaction representation switched back into the laboratory frame with inverse of (29-4). Now we show the vality of (29-5) by carrying out the differential. The left hand side of the equation is first expanded.

$$i\frac{d}{dt}\hat{\sigma} = i\frac{d}{dt}\{e^{iH_It}\sigma e^{-iH_It}\} = i\left\{(iH_I)e^{iH_It}\sigma e^{-iH_It} + e^{iH_It}\frac{d\sigma}{dt}e^{-iH_It} - e^{iH_It}\sigma(iH_I)e^{-iH_It}\right\}$$

$$i\frac{d}{dt}\hat{\sigma} = ie^{iH_It}\left\{(iH_I)\sigma + \frac{d\sigma}{dt} - \sigma(iH_I)\right\}e^{-iH_It} = ie^{iH_It}\left\{\frac{d\sigma}{dt} + i[H_I, \sigma]\right\}e^{-iH_It}$$

Equating this to the left hand side of the equation we obtain

$$ie^{iH_It}\left\{\frac{d\sigma}{dt} + i[H_I, \sigma]\right\}e^{-iH_It} = [\hat{\boldsymbol{H}}_{II}(t), \hat{\sigma}] \quad .$$

Now we can proceed to isolate the differential of the density matrix in the laboratory frame and expand the double commutator.

$$i\left\{\frac{d\sigma}{dt} + i[H_I, \sigma]\right\} = e^{-iH_It}[\hat{\boldsymbol{H}}_{II}(t), \hat{\sigma}]e^{iH_It} = [\boldsymbol{H}_{II}(t), \sigma]$$

$$i\frac{d\sigma}{dt} = [\boldsymbol{H}_{II}(t), \sigma] + [H_I, \sigma] = [H_I + \boldsymbol{H}_{II}(t), \sigma]$$

This being identical with the original Liouville equation, we have show that (29-5) is identical to (29-3).

## 4.4 Liouville Equation, Rotating Frame

Another common occurance in NMR has the acting Hamiltonian consisting of a time independent component as well as a time dependent component which is smoothly rotating about a specified axis. In such a case it is more beneficial to switch, not into the interaction representation, but into a coordinate system which rotates along with the time dependent Hamiltonian. Such a switch into the rotating frame can simplify solution of the Liouville equation by effectively removing the time dependence from Hamiltonian completely[1]. A second use of the rotating frame is NMR to remove some large overall frequecy, such as the Larmor frequency, so that all system transitions are seen on a Hertz rather than Megahertz scale[2]. We now consider again the switch into a coordinate system which is rotating about some axis $u$ at a frequency $\Omega$ . Generally, any operator can be switched

1. This is exactly what is needed when an external rf-field is applied to the system. Since rf is normally applied in the plane transverse to the direction of the static magnetic field, the xy-plane, it can be seen as oscillating about the z-axis and made to look time independent in the frame rotating about this axis at the rf-field frequency.

into this frame according to

$$\underset{\sim}{O} = R_u(\Omega t) O R_u^{-1}(\Omega t) = e^{-iF_u\Omega t} O e^{iF_u\Omega t} \tag{29-6}$$

where an underscore tilde, ~, is used to designate the rotating frame. What will the Liouville equation look like in this rotating frame? We can determine it directly by expanding the differential of the density matrix with respect to time in the rotating frame.

$$i\frac{d}{dt}\underset{\sim}{\sigma} = i\frac{d}{dt}\{e^{-iF_u\Omega t}\sigma e^{iF_u\Omega t}\} = i\frac{d}{dt}\{R_u(\Omega t)\sigma R_u^{-1}(\Omega t)\}$$

$$i\frac{d}{dt}\underset{\sim}{\sigma} = i\left\{(-iF_u\Omega)e^{-iF_u\Omega t}\sigma e^{iF_u\Omega t} + e^{-iF_u\Omega t}\frac{d\sigma}{dt}e^{iF_u\Omega t} + e^{-iF_u\Omega t}\sigma(iF_u\Omega)e^{iF_u\Omega t}\right\}$$

$$i\frac{d}{dt}\underset{\sim}{\sigma} = i\left\{(-iF_u\Omega)\underset{\sim}{\sigma} + R_u(\Omega t)\frac{d\sigma}{dt}R_u^{-1}(\Omega t) + \underset{\sim}{\sigma}(iF_u\Omega)\right\}$$

$$i\frac{d}{dt}\underset{\sim}{\sigma} = i\left\{R_u(\Omega t)\frac{d\sigma}{dt}R_u^{-1}(\Omega t) - i[F_u\Omega, \underset{\sim}{\sigma}]\right\}$$

We know from our original Liouville equation what the differential in the laborator frame is.

$$i\frac{d}{dt}\underset{\sim}{\sigma} = R_u(\Omega t)[\boldsymbol{H}(t), \sigma]R_u^{-1}(\Omega t) + [F_u\Omega, \underset{\sim}{\sigma}] \quad .$$

$$i\frac{d}{dt}\underset{\sim}{\sigma} = [\underset{\sim}{\boldsymbol{H}}(t), \underset{\sim}{\sigma}] + [F_u\Omega, \underset{\sim}{\sigma}] = [\underset{\sim}{\boldsymbol{H}}(t) + F_u\Omega, \underset{\sim}{\sigma}]$$

Thus we obtain the Liouville equation in the rotating frame.

$$i\frac{d}{dt}\underset{\sim}{\sigma} = [\underset{\sim}{\boldsymbol{H}}(t) + F_u\Omega, \underset{\sim}{\sigma}] \tag{29-7}$$

Like the solution in the interaction representation[1], the hope is that equation (29-7) is easier to solve than (29-3) and once accomplished the density matrix in the rotating frame switched back into the laboratory frame with inverse of (29-6).

2. This mimics using a reference or carrier frequency in an experiment. As a result, all frequencies are seen relative to the carrier. Again, since the Larmor precession of all magnetization occurs about the z-axis (the static field axis) we can switch into a frame rotating about this axis at a frequency near the Larmor frequecy to "referece" all measured from that value.
1. There are some authors who call this the interaction representation, I assume because $-F_z\Omega \approx H_0$ in homonuclear systems.

## 4.5 Liouville Equation, Multiple Rotating Frame

We can take the previous discussion concerning the rotating frame a step farther. In heteronuclear systems it is preferable to switch into a suitable reference frame for each isotope type in the system. That is to say, one wishes to maintain a different reference for each of the differing Larmor frequencies in a system. This presents no complications whatsoever as long as the multiple frames are rotationg about the same axis. We simply define selective rotation operator $R_{u,\{i\}}$ which affects only the set of spins $\{i\}$ of a particular isotope type.

$$\underset{\sim}{O}' = R_{u,\{i\}}(\Omega_{\gamma_i}t)OR^{-1}_{u,\{i\}}(\Omega_{\gamma_i}t) = e^{-iF_{u,\{i\}}\Omega_{\gamma_i}t}Oe^{iF_{u,\{i\}}\Omega_{\gamma_i}t}$$

In this equation we used the definition

$$F_{u,\{i\}} = \sum_{j}^{spins} \delta_{\gamma_i\gamma_j} I_{u,j}$$

so that the sum only involves spins sharing the same gyromagnetic ratio. A prime was used to indicate a modified rotating frame and a subscript of $\gamma_i$ to designate that the frequency $\Omega_{\gamma_i}$ is used for all spins of $\{i\}$ having the same gyromagnetic ratio. The Liouville equation in this selective rotating frame is derived the same as before.

$$i\frac{d}{dt}\underset{\sim}{\sigma}' = [\underset{\sim}{\boldsymbol{H}}'(t) + F_{u,\{i\}}\Omega_{\gamma_i}, \underset{\sim}{\sigma}']$$

There is nothing to prevent us from performing another selective rotation about the axis $u$ at a different frequency and different selectivity. The two applicable equations are

$$i\frac{d}{dt}\underset{\sim}{\sigma}'' = [\underset{\sim}{\boldsymbol{H}}''(t) + F_{u,\{i\}}\Omega_{\gamma_i} + F_{u,\{j\}}\Omega_{\gamma_j}, \underset{\sim}{\sigma}'']$$

and

$$\underset{\sim}{O}'' = R_{u,\{j\}}(\Omega_{\gamma_j}t)O'R^{-1}_{u,\{j\}}(\Omega_{\gamma_j}t)$$

Since all the selective rotation operators commute (they are applied about the same axis) the have no effect on each other and the order in which they are applied is inconsequential. For as many rotating frames $f$ we care to choose

$$i\frac{d}{dt}\underset{\sim}{\sigma} = \left[\underset{\sim}{\boldsymbol{H}}(t) + \sum_f F_{u,\{i_f\}}\Omega_f, \underset{\sim}{\sigma}\right]$$

we can write down the appropriate Liouville equation. The multiple rotating frame is given by

$$\underset{\sim}{O} = \prod_f [R_{u,\{i_f\}}(\Omega_f t)] O \prod_f [R^{-1}_{u,\{i_f\}}(\Omega_f t)] \quad .$$

and the nomeclature gets relatively messy. The end result is that we can work in as many rotating frames as we choose so long as they are rotating about the same axis.

## 4.6 Liouville Equation Solution, Static Hamiltonian

It is easy to verify that the solution to the Liouville equation under a time independent Hamiltonian is given by

$$\sigma(t) = e^{-i\boldsymbol{H}t}\sigma_o e^{i\boldsymbol{H}t} \; . \tag{29-8}$$

Here t is the time during which the Hamiltonian has acted on the spin system, $\sigma_o$ represents the initial state of the system and $\sigma(t)$ the final state. An alternative version of this equation, perhaps more clear, is

$$\sigma(t) = \sigma(t_0 + t_e) = e^{-i\boldsymbol{H}t_e}\sigma(t_0)e^{i\boldsymbol{H}t_e} \quad . \tag{29-9}$$

where the total time $t = t_0 + t_e$, $t_0$ is some initial time, and $t_e$ is the time over which the density matrix evolves under the effects of the Hamiltonian.

It is common to write this solution in terms of the propagator $\boldsymbol{U}$,

$$\sigma(t) = \boldsymbol{U}\sigma_o\boldsymbol{U}^{-1} \qquad \text{where} \qquad \boldsymbol{U} = e^{-i\boldsymbol{H}t} \tag{29-10}$$

and the time over which the propagator is active in implicit in the propagator itself[1]. It is enlightening to examine how the individual elements of $\sigma(t)$ behave under a time independent Hamiltonian. From (29-10) we have,

$$\langle\alpha|\sigma(t_e + t_0)|\beta\rangle = \sum_{\gamma}^{dim}\sum_{\gamma'}^{dim}\langle\alpha|\boldsymbol{U}|\gamma\rangle\langle\gamma|\sigma(t_0)|\gamma'\rangle\langle\gamma'|\boldsymbol{U}^{-1}|\beta\rangle \quad .$$

If the density matrix and propagators are in the basis of the static Hamiltonian, $\boldsymbol{H}$, then the propagator elements will be zero unless they are diagonals.

$$\langle\alpha|\sigma(t_e + t_0)|\beta\rangle = \langle\alpha|\boldsymbol{U}|\alpha\rangle\langle\alpha|\sigma(t_0)|\beta\rangle\langle\beta|\boldsymbol{U}^{-1}|\beta\rangle = \lambda_\alpha\lambda^*_\beta\langle\alpha|\sigma(t_0)|\beta\rangle$$

1. Under some circumstances it will be very important to know at which time the propagator is applied and the amount of time over which the propagator is active. We will deal with these on a case by case basis.

where $\lambda_\alpha$ are the eigenvalues of $\boldsymbol{U}$ which in turn are the exponentials of the eigenvalues of $\boldsymbol{H}$.

$$\lambda_\alpha^{\boldsymbol{U}} = \langle\alpha|\boldsymbol{U}|\alpha\rangle = \langle\alpha|e^{-i\boldsymbol{H}t}|\alpha\rangle = e^{-i\omega_\alpha t}$$

In this equation, the frequencies $\omega_\alpha$ are associated with the energy levels of $\boldsymbol{H}$. The end result is that each element of $\sigma$ (or any other operator evolving under the effects of $\boldsymbol{H}$) oscillates in the complex plane at a specific transition frequency as given by

$$\langle\alpha|\sigma(t_e + t_0)|\beta\rangle = e^{-i\omega_\alpha t}e^{i\omega_\beta t}\langle\alpha|\sigma(t_0)|\beta\rangle = e^{-i\omega_{\alpha\beta}t}\langle\alpha|\sigma(t_0)|\beta\rangle \quad (29\text{-}11)$$

where $\omega_{\alpha\beta} = \omega_{\alpha\beta} - \omega_{\alpha\beta}$.

We can also write down the equivalent to equation (29-10), the solution to the von Neumann equation under a time independent Hamiltonian, in superoperator formalism as[1]

$$\sigma(t + t_0) = e^{-i\boldsymbol{H}t}\sigma(t_0)e^{i\boldsymbol{H}t} = \boldsymbol{U}\sigma(t_0)\boldsymbol{U}^{-1} = U\sigma(t_0) \quad . \quad (29\text{-}12)$$

In this context, the superoperator equivalent of the unitary transformation involving the propagator $\boldsymbol{U}$, namely $U$, is determined from[2]

$$U = \boldsymbol{U} \otimes \boldsymbol{U}^*$$

Where $\boldsymbol{U}^*$ is the complex conjugate of the propagator $\boldsymbol{U}$ and $\otimes$ is a tensor product.

1. See EBW, page 16, equation (2.1.41)
2. See EBW, page 24, equation (2.1.83).

## 4.7 Example Source Codes