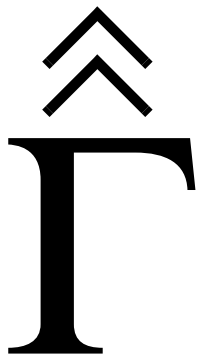
GAMMA

Decoupling Examples



Author: Scott A. Smith

Date: March 15, 2000

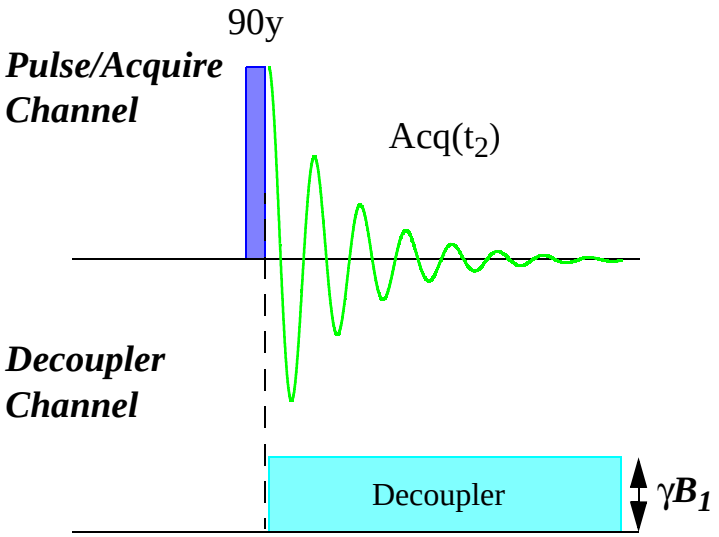
Table of Contents

1 Introduction

1.1 Overview

This document demonstrates how users may introduce decoupling into their NMR simulations. It assumes that the reader is familiar with the general facets of GAMMA based simulations and will not cover details such as spin system definitions or the use of operators and superoperators. Rather, the majority of this text will focus on the use of decoupling steps in a pulse sequence, such as that shown in the figure below.

Basic Decoupling During An Acquisiton



Decoupling steps may be introduced on any specified channel and they can be used multiple times in the same simulation, during either a set delay and/or an acquisition.

Please note that the examples programs contained in this book are meant as simple tutorials in getting your own 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 decoupling programs see the GAMMA WWW site at http::/gamma.magnet.fsu.edu.

1.2 Decoupling Types

Decoupling is of course used to remove scalar coupling(s) during one or more steps in an NMR pulse sequence. Experimentally, this is accomplished using applied rf-irradiation. In GAMMA, the simplest way to remove scalar couplings during a pulse sequence step is to evolve the spin system

under a (modified) Hamiltonian which has had such coupling terms removed. The result will then be “perfect” decoupling. Often this will be the optimal decoupling method to use because it is simple to implement and produces most of the desired effect. This type of decoupling will be covered in the next chapter and herein called “ideal decoupling”.

However, when running decoupling experiments such perfection is never achieved. There will always be some additional “imperfections” introduced into the experimental results which arise from the applied rf-field itself. These will include effects from the decoupler strength, phase, and offset frequency. Users may require that such details are included in a simulation, and such provisions have been made in GAMMA.

The simplest decoupling to use (and still account for applied rf-field effects) is CW (continuous wave) decoupling. In such instances rf-irradiation is applied on the channel to be decoupled during an evolution in a pulse sequence. The user specifies the decoupler strength (gB1) as well as the rf-field phase and frequency offest. This type of simulation will be covered in the chapter “CW decoupling”.

To improve decoupling efficiency, there are a wide variety of pulse trains which are used above simple CW decoupling. These normally work (decouple) over a broader range of frequencies and require less rf-power to attain the same effect. GAMMA provides some of these decoupling schemes (MLEV, WALTZ, GARP, CHIRP) as well as a means for the user to generate their own. These will be covered in chapters bearing the name of the decoupling sequence.

Additional details that will be covered in this book are how the effects of relaxation can be included during decoupling.

2 Ideal Decoupling

2.1 Introduction

Ideal decoupling is “perfect” decoupling. That is, the system evolves without the presence of the “decoupled” J (scalar coupling) interactions. There is nothing mysterious about the GAMMA calculations. Without decoupling the system evolves under one Hamiltonian, with decoupling the system evolves under a modified Hamiltonian - ***modified to neglect the decoupled scalar interactions***.

Why in the world would one choose ideal coupling? Because it is the easiest to use! Perhaps you are doing a complex simulation and you don’t want to worry about your decoupling steps (at least not for starters). Perhaps you want a program that illustrates the basics of decoupling without the hassle of offset effects, decoupler power, etc. Perhaps you wish to compare your own complicated decoupling sequence’s performance to what the ideal decoupling result would be. All of these are compelling reasons to use ideal decoupling.

The rest of this chapter provides direct examples of ideal decoupling. Please note that these examples are meant as simple tutorials in getting decoupling 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 decoupling programs see the GAMMA WWW site at http://gamma.magnet.fsu.edu.

2.2 Decoupling During Acquisitions

For our first example we'll begin with a very simple simulation, a decoupled NMR spectrum. We'll apply a hard 90 pulse on one channel then detect the signal while decoupling on another channel. Remember, we won't be doing anything complicated, just removing the scalar coupling terms directly from the Hamiltonian which is active during decoupling. Here is the pulse sequence we'll be simulating:

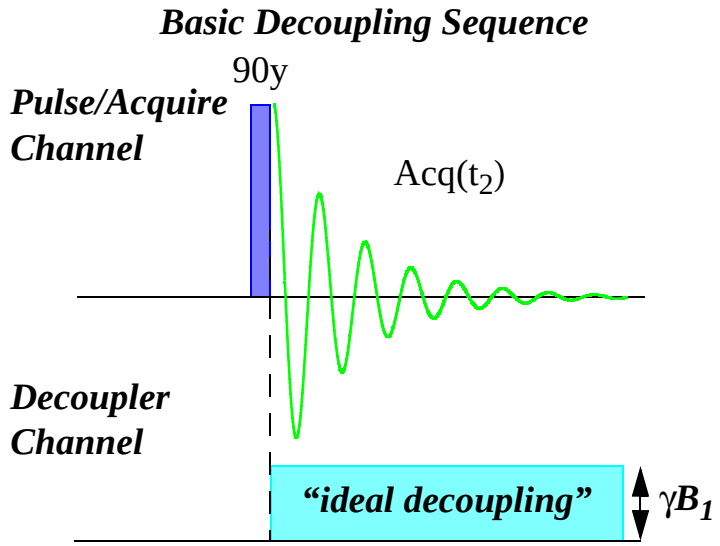
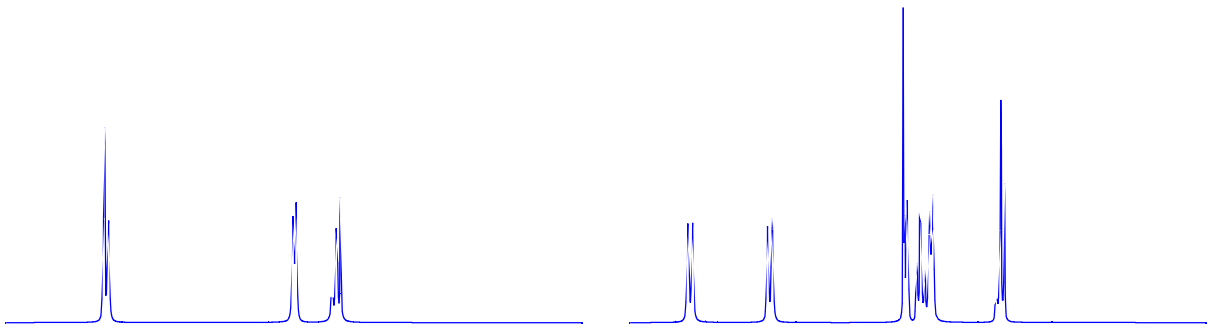


Figure 0-1 **A simple decoupling pulse sequence.**

The following figure contains two GAMMA simulated spectra generated for this pulse sequence. The figure on the left is not decoupled whereas the on on the right is.



The source code for our ideal decoupling program is given on the following page. It is simply a GAMMA 1D NMR simulation program except for some minor details. 1.) It requests pulse and decoupler channels (if the system is heteronuclear) and 2.) It uses a "decoupled" isotropic Hamiltonian rather than the normal high-resolution NMR Hamiltonian.

```
/* IdealDecGP0.cc ***************************************************-*-c++-*-
**                                                                       **
**              Example Program for the GAMMA Library                    **
**                                                                       **
**  This program calculates a 1D spectrum using ideal decoupling.  It    **
**  first applies a hard 90 pulse to the detection channel then evolves  **
**  without the supressed scalar coupling (which is decoupled) during    **
**  acqusiiton.  no relaxation effects are considered during any steps.  **
**                                                                       **
**  This program runs interactively, asking the user to supply the       **
**  spin system filename, the number of t2 points desired, the isotope   **
**  channels for pulse/detection and decoupling, and plot parameters.    **
**  Spectral output is produced in Gnuplot format and plotted inter-     **
**  actively if gnuplot is known to the system.                          **
**                                                                       **
**  Assuming a.out is the executable, you should be able to obtain a     **
**  13C decoupled proton spectrum using the command                      **
**                                                                       **
**  a.out IdealDecGP0.sys 1024 1H 13C 1.0                                **
**                                                                       **
**  then answering the question about the Nyquist frequency with a "u".  **
**  The coupling can be left intact if a non-existent isotope is chosen  **
**  for the decoupler channel, i.e.                                      **
**                                                                       **
**  a.out IdealDecGP0.sys 1024 1H 19F 1.0                                **
**                                                                       **
***************************************************************************/

#include <gamma.h>

main (int argc, char* argv[])
 {
 cout << "\n\tGAMMA Decoupling Simulation: Gnuplot, No Relaxation\n";
//                    Read in Parameters
 int qn = 1;                                  // Parameter query number
 spin_system sys;                             // A spin system
 sys.ask_read(argc,argv,qn++);                // Read in/Ask for system
 cout << sys;                                 // Have a look at the system
 int t2pts;                                   // Block size
 query_parameter(argc, argv, qn++,            // Get number FID of points
         "\n\tAcquisition Size? ", t2pts);
 String IsoD, IsoDec;                         // Pulse/Detect, Decouple
 if(sys.homonuclear())                        // Set pulse/detect and
  {                                           // decoupling channels the
  IsoD = sys.symbol(0);                       // same if homonuclear
  IsoDec = IsoD;                              // system input
  }
 else                                         // For heteronuclear systems
  {                                           // Ask for the channel types
  query_parameter(argc, argv, qn++,           // e.g. 1H, 13C, 19F, ...
      "\n\tPulse/Detect Channel? ", IsoD);
  query_parameter(argc, argv, qn++,
      "\n\tDecoupling Channel? ", IsoDec);
  }

 double lwhh = 3.0;                           // Half-height linewidth
 query_parameter(argc, argv, qn++,            // Ask for apodization strength
         "\n\tApodization (Hz)? ", lwhh);
//                    SetVariables
 gen_op H = Hcs(sys) + HJd(sys, IsoDec);      // Set the decoupled Hamiltonian
 gen_op detect = Fm(sys, IsoD);               // Set detection operator to F-
 block_1D fid(t2pts);                         // A block for acquisition
//                    Set Up Spectral Parameters
 double NyqF = query_Nyquist(sys, IsoD);      // Choose a Nyquist frequency
 double dt = 1.0/(2.0*NyqF);                  // Dwell time, quadrature
 double SW = 2.0*NyqF;                        // Total Spectal width +/- Nyquist
//                    Implement Pulse - Acquisition Sequence
 gen_op sigma0 = sigma_eq(sys);               // Start at equilibrium
 gen_op sigma1 = Iypuls(sys, sigma0, 90);     // Apply (PI/2)y ideal pulse
 FID(sigma1,detect,H,dt,t2pts,fid);           // Calculate FID under H
//                    Process and Output Data
 double RR = (lwhh/2)*HZ2RAD;                 // Set apodization rate
 double tt = double(t2pts-1)*dt;              // Total FID length
 row_vector vex=Exponential(t2pts,tt,0.0,RR,0); // Block for apodization
 row_vector  fidap = product(fid,vex);        // Apodized the FID
 row_vector data = FFT(fidap);                // Apply FFT
 cout << "\n\n";                             // Keep screen nice
 cout.flush();                                // Insure writing all done
 GP_1D("data.asc", data, 0, -(SW/2), (SW/2)); // Output gnuplot ASCII file
 ofstream gnuload("data.gnu");                // File of gnuplot commands
 gnuload << "set data style line\n";          // Set 1D plots to use lines
 gnuload << "set xlabel \"W2(Hz)\"\n";        // Set X axis label
 gnuload << "set ylabel \"Intensity\"\n";     // Set Y axis label
 gnuload << "set title\"Spectra\"\n";         // Set plot title
 gnuload << "plot \"data.asc\"\n";            // Plot FIDs in gnuplot
 gnuload << "pause -1 \'<Return> To Exit \n"; // Pause before exit
 gnuload << "exit\n";                         // Now exit gnuplot
 gnuload.close();                             // Close gnuplot command file
 system("gnuplot \"data.gnu\"\n");            // Plot to screen
 cout << "\n\n";                             // Keep the screen nice
 }
```

The input spin system which produced the previously shown spectra is given below.

SysName	(2) : CHdec	- Name of the Spin System
NSpins	(0) : 4	- Number of Spins in the System
Iso(0)	(2) : 13C	- Spin Isotope Type
Iso(1)	(2) : 1H	- Spin Isotope Type
Iso(2)	(2) : 1H	- Spin Isotope Type
Iso(3)	(2) : 1H	- Spin Isotope Type
v(0)	(1) : 100.0	- Chemical Shifts in Hz
v(1)	(1) : -87.0	- Chemical Shifts in Hz
v(2)	(1) : 392.0	- Chemical Shifts in Hz
v(2)	(1) : -202.0	- Chemical Shifts in Hz
J(0,1)	(1) : 166	- Coupling Constants in Hz
J(0,2)	(1) : 166	- Coupling Constants in Hz
J(0,3)	(1) : 52.7	- Coupling Constants in Hz
J(1,2)	(1) : -10.0	- Coupling Constants in Hz
J(1,3)	(1) : 7.0	- Coupling Constants in Hz
J(2,3)	(1) : 3.0	- Coupling Constants in Hz
Omega	(1) : 500	- Spect. Freq. in MHz (1H)

This was fed into the compiled program. For the decoupled spectrum the program command line read

```
a.out IdealDecGP0.sys 1024 1H 13C 1
```

and for the spectrum with no decoupling

```
a.out IdealDecGP0.sys 1024 1H 19F 1
```

Note that I did set the requested Nyquist frequency to 600 in both cases so that both plots would be put on the same horizontal scale.

For emphasis, this “decoupling” program is exactly the same as a 1D NMR simulator *except* for the fact that a “decoupled” Hamiltonian was used to evolve the system during the acquisition. The code line was highlighted in blue on the previous page.

Some final comments. The computation was performed in the time domain, requiring an FFT to produce the spectrum. It may, as demonstrated in other GAMMA examples using the class acquire1D, also be performed directly in the frequency domain. Also, the lines to produce a spectrum in Gnuplot can all be replaced by the simple commands

```
GP_1D(“data.asc”, data, 0, -(SW/2), (SW/2)); // Output gnuplot ASCII file
GP_1Dplot(“data.gnu”, “data.asc””);           // Plot to screen
```


2.3 Decoupling During Delays

This section details how to use “ideal” decoupling during a time delay. The easiest of these is to usedecoupled NMR spectrum. We’ll apply a hard 90 pulse on one channel then detect the signal while decoupling on another channel. Remember, we won’t be doing anything complicated, just removing the scalar coupling terms directly from the Hamiltonian which is active during decoupling. Here is the pulse sequence we’ll be simulating:

3 CW Decoupling

3.1 Introduction

Continuous Wave (CW) decoupling is also quite simple to implement in GAMMA programs. Similar to the “ideal” decoupling case, the problem can be formulated such that a “static” Hamiltonian is active while a constant rf-field is applied. The pulse sequence below depicts a simple CW decoupling experiment.

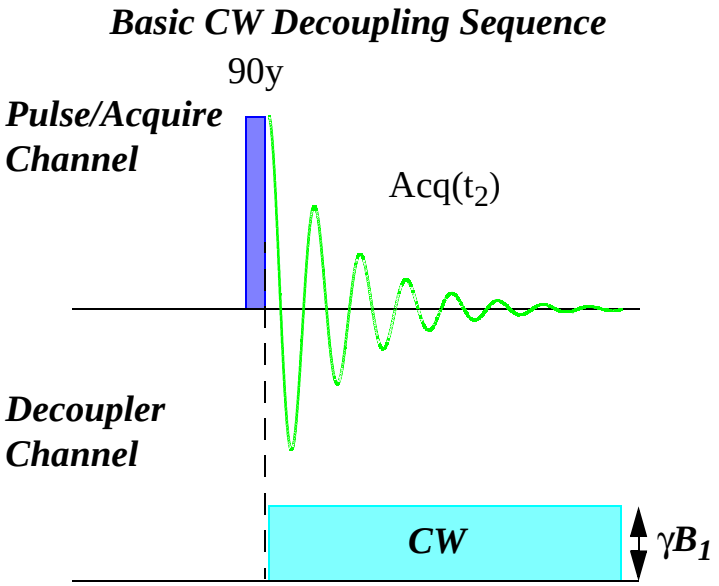


Figure 0-2 **A simple pulse-acquisition sequence to obtain a decoupled spectrum using CW decoupling.**

The first pulse generates magnetization in the xy-plane. Subsequently, the magnetization is detected during t_2 while an rf-field of constant amplitude, frequency and phase is applied. Scalar couplings involving spins which resonate near the applied rf frequency will be quenched.

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

3.2 Decoupling During Acquisitions

For our first example we'll begin with a very simple simulation, a decoupled NMR spectrum. The pulse sequence we will implement is shown in the following figure.

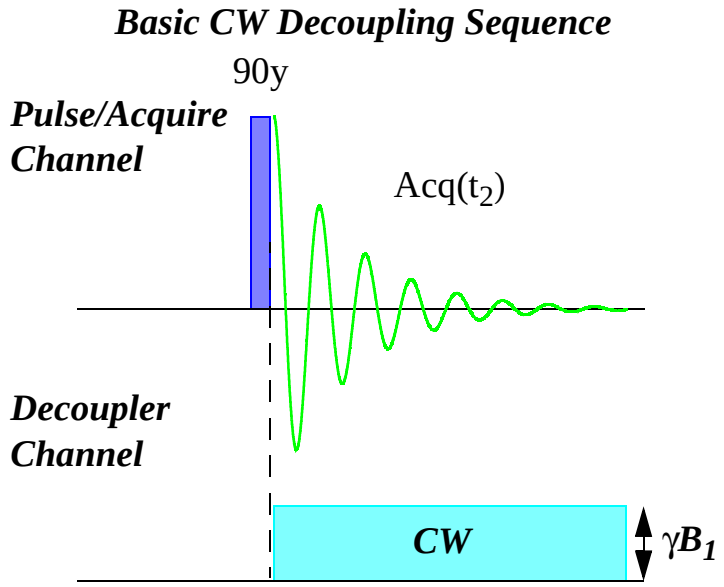


Figure 0-3 **A simple hetero-nuclear decoupling pulse sequence.**

We'll draw upon our "ideal" decoupling simulation (previous chapter) and simply replace the active Hamiltonian to include all scalar couplings as well as a term for the applied rf-field. The important part of this simulation is that it must be performed in the rotating frame of the applied rf-field. That is, in order for our Hamiltonian to remain constant we must be in a rotating frame with the applied field.

3.3 CWdecGP0.cc

CWdecGP0.cc

```
/* CWdecGP0.cc ***************************************************************-*-c++-*-
**                                                                             **
**  This program calculates a 1D spectrum during CW decoupling.  It is         **
**  quite simple in that it first applies a hard 90 pulse to the               **
**  detection channel and no relaxation effects are considered during          **
**  the acquisition.                                                           **
**                                                                             **
**  Assuming a.out is the executable, you should be able to obtain a           **
**  13C decoupled proton spectrum using the command                            **
**                                                                             **
**            a.out CWdecGP0.sys 1024 1H 13C 5000 1.0                          **
**                                                                             **
**  then answering the question about the Nyquist frequency with a "u".        **
**                                                                             **
** Author:    S.A. Smith                                                       **
***                                                                           **
*********************************************************************************/

#include <gamma.h>

main (int argc, char* argv[])
 {
 cout << "\n\tGAMMA Decoupling Simulation: Gnuplot, No Relaxation\n";
//                              Read in Parameters
 int qn = 1;                                    // Parameter query number
 spin_system sys;                               // A spin system
sys.ask_read(argc,argv,qn++);                   // Read in/Ask for system
 cout << sys;                                   // Have a look at the system
 int t2pts;                                     // Block size
 query_parameter(argc, argv, qn++,              // Get number FID of points
          "\n\tAcquisition Size? ", t2pts);
 String IsoD, IsoCW;                            // Pulse/Detect, Decouple
 if(sys.homonuclear())                          // Set pulse/detect and
  {                                             // decoupling channels the
  IsoD = sys.symbol(0);                         // same if homonuclear
  IsoCW = IsoD;                                 // system input
  }
 else                                           // For heteronuclear systems
  {                                             // Ask for the channel types
  query_parameter(argc, argv, qn++,             // e.g. 1H, 13C, 19F, ...
     "\n\tPulse/Detect Channel? ", IsoD);
  query_parameter(argc, argv, qn++,
       "\n\tDecoupling Channel? ", IsoCW);
  }
 double gamB1;
 query_parameter(argc, argv, qn++,              // Get the decoupling strength
          "\n\tDecoupling Field Strength[Hz]? ", gamB1);
 double lwhh = 3.0;                             // Half-height linewidth
 query_parameter(argc, argv, qn++,              // Ask for apodization strength
          "\n\tApodization (Hz)? ", lwhh);
//                              Set Variables
 gen_op H = Ho(sys);                            // Set isotropic Hamiltonian
 gen_op Heff = H + gamB1*Fx(sys,IsoCW);         // Set the effective Hamiltonian
 gen_op detect = Fm(sys, IsoD);                 // Set detection operator to F-
 block_1D fid(t2pts);                           // A block for acquisition
//                          Set Up Spectral Parameters
 double NyqF = query_Nyquist(sys, IsoD);        // Choose a Nyquist frequency
 double dt = 1.0/(2.0*NyqF);                    // Dwell time, quadrature
 double SW = 2.0*NyqF;                          // Total Spectal width +/- Nyquist
//                      Implement Pulse - Acquisition Sequence
 gen_op sigma0 = sigma_eq(sys);                 // Start at equilibrium
 gen_op sigma1 = Iypuls(sys, sigma0, 90);       // Apply (PI/2)y ideal pulse
 FID(sigma1,detect,Heff,dt,t2pts,fid);          // Calculate FID under Heff
//                          Process and Output Data
 double RR = (lwhh/2)*HZ2RAD;                   // Set apodization rate
 double tt = double(t2pts-1)*dt;                // Total FID length
 row_vector vex=Exponential(t2pts,tt,0.0,RR,0); // Block for apodization
 row_vector  fidap = product(fid,vex);          // Apodized the FID
 row_vector data = FFT(fidap);                  // Apply FFT
 cout << "\n\n";                               // Keep screen nice
 cout.flush();                                  // Insure writing all done
 GP_1D("data.gnu", data, 0, -(SW/2), (SW/2));   // Output gnuplot ASCII file
 ofstream gnuload("gnu.dat");                   // File of gnuplot commands
 gnuload << "set data style line\n";            // Set 1D plots to use lines
 gnuload << "set xlabel \"W2(Hz)\"\n";          // Set X axis label
 gnuload << "set ylabel \"Intensity\"\n";       // Set Y axis label
 gnuload << "set title\"Spectra\"\n";           // Set plot title
 gnuload << "plot \"data.gnu\"\n";              // Plot FIDs in gnuplot
 gnuload << "pause -1 \'<Return> To Exit \n";   // Pause before exit
 gnuload << "exit\n";                           // Now exit gnuplot
 gnuload.close();                               // Close gnuplot command file
 system("gnuplot \"gnu.dat\"\n");               // Plot to screen
 cout << "\n\n";                               // Keep the screen nice
 }
```

How does this account for the rf-field rotating frame? It actually works in multiple rotating frames, one for each isotope type. Only the decoupler channel is in the rotating frame of the field, the rest have all of their assocated spins referenced to some other frame. Note that this is an approximation! It assumes that there are no re-

sidual effects from working in such frame and throws away the corresponding time dependent Hamiltonian terms[1].

1. In isotropic systems the active Hamiltonian is taken as a combination of Zeeman and scalar coupling terms. It is the latter which becomes time-dependent when switched into a multiple rotating frame. However, if the rotating frames are far apart such terms are negligible. This will almost always be the case since the rotating frames are separated by the isotope Larmor frequency differences. Exceptions will be when the Bo field is turned down and/or when working with heteronuclei which happen tohave similar Larmor frequencies. In that rare event one is forced to work in a single rotating frame (but that isn't really a problem then anyway). A much worse situation occurs when one attempts to work in a multiple rotating frame on the same channel because the time-dependent terms are then NOT small.

3.4 Homonuclear Decoupling During Acquisitions

For our first example we'll begin with a very simple simulation, a decoupled NMR spectrum.

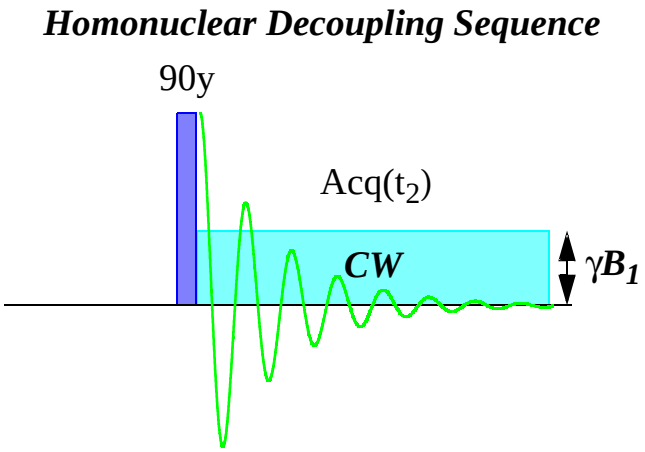


Figure 19-4 A simple decoupling pulse sequence .

3.5 CW Decoupling Profile

Now we'll generate a profile. Such experiments are usually done in order to evaluate decoupling performance, typically for some multiple pulse decoupling scheme. We'll do one for CW decoupling so it won't be anything new when encountered in some broad-band decoupling simulation later in this book. The pulse sequence we will implement is the same one we've used before, shown in the following figure.

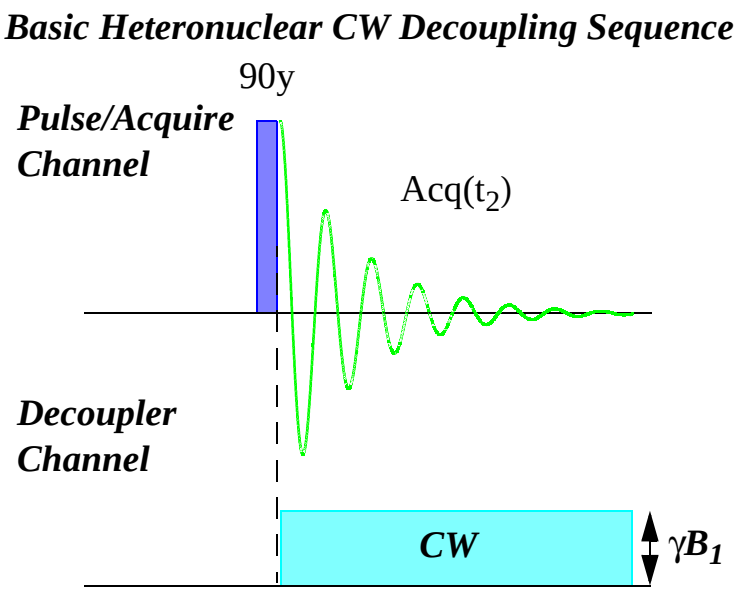


Figure 19-5 **A simple hetero-nuclear decoupling pulse sequence**

.

The experiment is normally performed on a 2-spin heteronuclear spin system. A typical sample (ala Freeman and Shaka) would be ^{13}C labeled formate where decoupling is applied on the carbon channel and the aldehydic proton is detected. The decoupler rf offset is changed with each repeated experiment and the proton spectra from these placed side by side.

First, let us pick a simple spin system. The file below, named CWdecprof0.sys, will be used as input. You can't get any simpler than this.

CWdecprof.sys

```
SysName   (2) : CHprof    - Name of the Spin System
NSpins    (0) : 2         - Number of Spins in the System
Iso(0)    (2) : 13C       - Spin Isotope Type
Iso(1)    (2) : 1H        - Spin Isotope Type
v(0)      (1) : 0.0       - Chemical Shifts in Hz
v(1)      (1) : 0.0       - Chemical Shifts in Hz
J(0,1)    (1) : 166       - Coupling Constants in Hz
Omega     (1) : 500       - Spect. Freq. in MHz (1H)
```

Note that both the carbon and proton are set to be on resonance. For the proton this means that our generated spectra will be symmetric about 0 Hz. For carbon we want to start at 0 Hz so we can reference out decoupler rf offset. Since we are not including relaxation, and the scalar coupling is weak, the Bo field strength will have no influence on our results.

Now lets have a look at the code itself. We will modify one of the previous programs to ask the user for information pertaining to the profile then add a loop over the offset values. The only "new" aspect here is that the spectra are placed into a single block (row vector). All of this I/O and looping makes the program get a bit long....

CWdecGP0.cc

```
/* CWdecprof0.cc ************************************************************-*-c++-*-
**                                                                            **
**          GAMMA CW Decoupling Profile Simulation Program                    **
**                                                                            **
** This program simulates CW decoupling.  It produces a decoupling            **
** profile, i.e. a series of 1D plots stacked side to side where each         **
** plot is produced at a different decoupler offset. The user specifies       **
** the number of offsets and the offset increment.                            **
**                                                                            **
```

```
** The program is designed for a 2-spin heteronuclear system.  The user      **
** chooses the spin system and which channel to detect.  The pulse           **
** channel is set the same as the detection channel and the decoupler        **
** channel set to the hetero-nucleus.                                        **
**                                                                           **
** No relaxation is accounted for, the user must set a line-broadening       **
** for the acquired spectra.                                                 **
**                                                                           **
**  Assuming a.out is the executable, you should be able to obtain a         **
**  13C proton decouped profile using the command                            **
**                                                                           **
**          a.out CWdecprof0.sys 1H 2000 1000 10 1024 100 5                  **
**                                                                           **
** Author:    S.A. Smith                                                     **
** Date:      5/9/1996                                                       **
** Last Date: 4/17/98                                                        **
** Copyright: S.A. Smith, September 1995                                     **
** Limits:    1.) Needs >= GAMMA 3.5                                         **
**            2.) Output is gnuplot interactive.                             **
**            3.) Assumes an isotropic spin system.                          **
**            4.) Relaxation effects are included.                           **
**                                                                           **
*******************************************************************************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA CW Decoupling Profile Simulation\n";
//                          Read in Parameters
 int qn=1;
 spin_system sys;                                   // Declare dynamic system sys
 sys.ask_read(argc,argv,qn++);                      // Ask for file name of sys
 if(sys.spins() !=2 || sys.homonuclear())           // Insure proper input system
  {
  cout << "\n\tSorry, this program is designed to take a two"
       << " spin heteronuclear spin system ONLY!\n\n";
  exit(-1);
  }
 String IsoD, IsoCW;                                // Detector, Decoupler channel
 query_parameter(argc, argv, qn++,
   "\n\tDetection Channel (e.g. 1H, 13C, ...)? ", IsoD);
 if(sys.symbol(0) == IsoD)      IsoCW = sys.symbol(1);
 else if(sys.symbol(1) == IsoD) IsoCW = sys.symbol(0);
 else
  {
  cout << "\n\tSorry, there are no spins of type " << IsoD
       << " present.  Try another detection channel!\n\n";
  exit(-1);
  }
//                      Set Decoupling Profile Parameters
 double gamB1;
 query_parameter(argc, argv, qn++,                  // Get the decoupling strength
  "\n\tDecoupling Field Strength[Hz]? ", gamB1);
 double SWprof;
 query_parameter(argc, argv, qn++,                  // Get the profile range
  "\n\tTotal Profile Spectral Width[Hz]? ", SWprof);
 int NO = 30;                                       // # Of Offsets (on each side)
 query_parameter(argc, argv, qn++,                  // Get # offsets
  "\n\tNumber of Positive Decoupler Offsets? ",
                                NO);
 double offset = SWprof/(2.0*double(NO));           // Set offset increment (Hz)
 double totaloff = -SWprof/2.0;                     // Starting offset value (Hz)
//                      Set Individual Spectrum Parameters
 int t2pts;
 query_parameter(argc, argv, qn++,                  // Get # offsets
  "\n\n\tNumber of Points Per Offset? ", t2pts);
 double SW;                                         // Nyquist frequency
 query_parameter(argc, argv, qn++,                  // Get # offsets
  "\n\n\tSpectral Width Per Offset? ", SW);
 double lwhh = 3.0;                                 // Half-height linewidth
 query_parameter(argc, argv, qn++,                  // Ask for apodization strength
          "\n\tApodization (Hz)? ", lwhh);
//                          Set Variables
 double dt = 1.0/(SW);                              // Dwell time, quadrature
 double RR = (lwhh/2)*HZ2RAD;                       // Set apodization rate
 double tt = double(t2pts-1)*dt;                    // Total FID length per offset
 row_vector vex=Exponential(t2pts,tt,0.0,RR,0);     // Block for apodization
//                      Output Specified Parameters
 cout << "\n\tSet FID Dwell Time To \t" << dt;
 cout << "\n\tSet Spectral Width To \t" << SW;
 cout << "\n\tSet LW @ Half-Height To \t" << lwhh;
 cout << "\n\tProfile Entry Points  \t" << t2pts;
 cout << "\n\tProfile Total Points  \t" << t2pts*(2*NO+1);
 cout << "\n";
 cout.flush();
 sys.offsetShifts(-NO*offset, IsoCW);
 block_1D bdata(t2pts);                             // Block for spectrum
 row_vector fidap;                                  // For apodized FID
 row_vector data(t2pts);                            // Block for spectrum
 row_vector profile((2*NO+1)*t2pts, complex0);      // Block for profile
 int K=0;
 String nam;
 String st="cwoff";
 String fi=".asc";
 double actoff = totaloff;                          // Current offset value (Hz)
 int len = 2;
 if(NO >9)   len++;
```

```
if(NO >99)  len++;
if(NO >999) len++;
String iform = String("%") + dec(len) + String("i");
for(int ov=-NO; ov<=NO; ov++)
 {
 cout << "\n\tSimulating Offset " << form(iform, ov)
      << " at " << form("%8.2f", actoff) << " Hz";
 cout.flush();
 gen_op H = Ho(sys);                          // Set isotropic Hamiltonian
 gen_op Heff = H + gamB1*Fx(sys,IsoCW);       // Set the effective Hamiltonian
 gen_op detect = Fm(sys, IsoD);               // Set detection operator to F-
 gen_op sigma0 = sigma_eq(sys);               // Set density mx equilibrium
 gen_op sigmap=lypuls(sys,sigma0,IsoD,90.0);  // Apply a 90 pulse
 FID(sigmap,detect,Heff,dt,t2pts,bdata);      // Calculate FID under Heff
 data = bdata;                                // Make it a row_vector
 fidap = product(data,vex);                   // Apodized FID this offset
 data = FFT(fidap);                           // From propagator time domain
 for(int k=0; k<t2pts; k++, K++)              // Store spectrum in profile
   profile.put(data.get(k), K);
 sys.offsetShifts(offset, IsoCW);             // Offset detection channel
 actoff += offset;                            // Track current offset value
 }

double Hzppt = SWprof/(2.0*double(NO));
double sumoff = Hzppt*double(2*NO+1)/2.0;
GP_1D("profile.gnu",profile,0,sumoff,-sumoff);  // Output spectrum in gnuplot
cout << "\n\n";                                 // Clean up screen
cout.flush();                                   // Flush output before gnuplot
ofstream gnuload("CW.gnu");                     // File of gnuplot commands
gnuload << "set data style line\n";             // Set 1D plots to use lines
gnuload << "set xlabel \"Offset(Hz)\"\n";       // Set X-axis label
gnuload << "plot \"profile.gnu\"\n";            // Plot the spectrum
gnuload << "pause -1 \'<Return> To Exit \n";    // Pause before quitting gnuplot
gnuload << "exit\n";                            // Exit gnuplot
gnuload.close();                                // Close gnuplot command file
system("gnuplot \"CW.gnu\"\n");                 // Plot to screen
FM_1D("CWprof.mif",profile,14,14,sumoff,-sumoff);  // FM 1D plot file
cout << "\n\n";                                 // Keep the screen nice
}
```

Now we can run the program. Assuming that we have compiled the program to produce an executable "a.out" (this will likely be a.exe on a Windoze system), we can run the program with the command

a.out CWdecprof0.sys 1H 2000 1000 10 1024 100 5

But one can also run the program interactively. Here is the dialog associated with such a run.

```
|gamma1>a.out

GAMMA CW Decoupling Profile Simulation
Spin system filename? CWdecprof0.sys
Detection Channel (e.g. 1H, 13C, ...)? 1H
Decoupling Field Strength[Hz]? 2000
Total Profile Spectral Width[Hz]? 1000
Number of Positive Decoupler Offsets? 10
Number of Points Per Offset? 1024
Spectral Width Per Offset? 200
Apodization (Hz)? 5

Set FID Dwell Time To           0.005
```

```
Set Spectral Width To          200
Set LW @ Half-Height To        5
Profile Entry Points           1024
Profile Total Points           21504

Simulating Offset -10 at  -500.00 Hz
Simulating Offset  -9 at  -450.00 Hz
Simulating Offset  -8 at  -400.00 Hz
Simulating Offset  -7 at  -350.00 Hz
Simulating Offset  -6 at  -300.00 Hz
Simulating Offset  -5 at  -250.00 Hz
Simulating Offset  -4 at  -200.00 Hz
Simulating Offset  -3 at  -150.00 Hz
Simulating Offset  -2 at  -100.00 Hz
Simulating Offset  -1 at   -50.00 Hz
Simulating Offset   0 at     0.00 Hz
Simulating Offset   1 at    50.00 Hz
Simulating Offset   2 at   100.00 Hz
Simulating Offset   3 at   150.00 Hz
Simulating Offset   4 at   200.00 Hz
Simulating Offset   5 at   250.00 Hz
Simulating Offset   6 at   300.00 Hz
Simulating Offset   7 at   350.00 Hz
Simulating Offset   8 at   400.00 Hz
Simulating Offset   9 at   450.00 Hz
Simulating Offset  10 at   500.00 Hz
```

The profile produced from this is shown in the figure below.

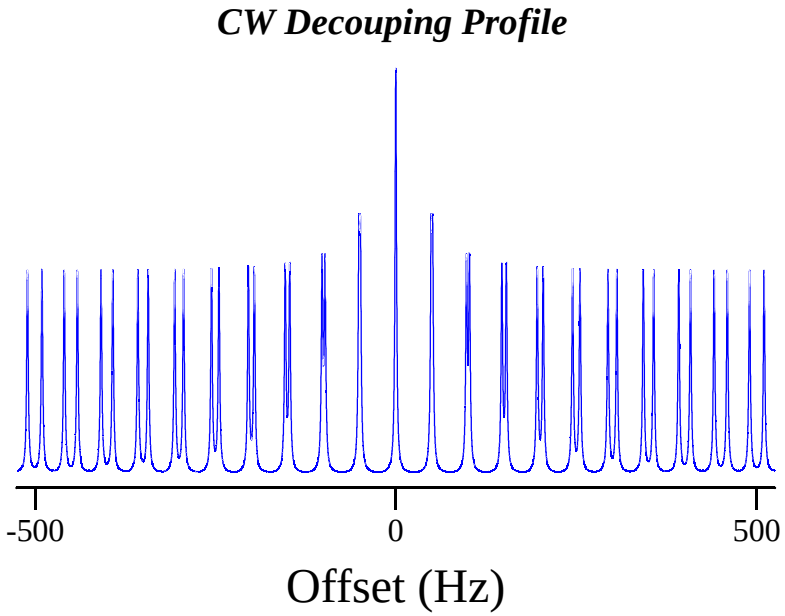


Figure 19-6 **CW decoupling profile genetared from CWdecprof0.cc.**

4 Multiple Pulse Decoupling

4.1 Introduction

Multiple pulse techniques such as MLEV, WALTZ, GARP, and CHIRP, have been developed to enhance decoupling performance beyond the CW decoupling (discussed in the previous chapter). In these instances the applied decoupling field will potentially be moduled in strength, offset, and phase. Due to the nature of such sequences, there is no constant Hamiltonian which is active during the decoupling. Consequently, we must mathmatically treat simulations involving such decoupling quite differently than has been discussed in previous sections. The pulse sequence below is representative of a simple multipulse decoupling experiment.

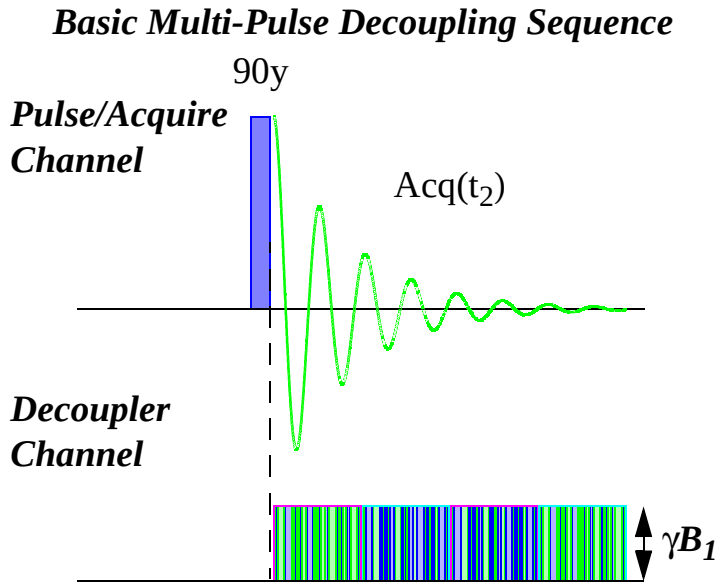


Figure 19-7 **A simple pulse-acquisition sequence involving a multiple pulse decoupling scheme. Keep in mind that the rf-amplitude, phase, and offset can vary on each step of the multiple pulse sequence.**

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.

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.

4.2 Heteronuclear Decoupling During Acquisitions

For our first example we'll begin with a very simple simulation, a decoupled NMR spectrum. The pulse sequence we will implement is shown in the following figure.

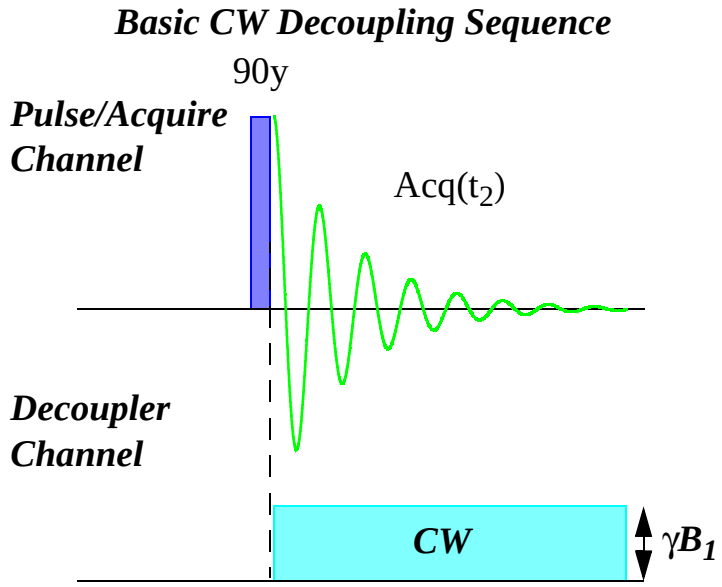


Figure 19-8 **A simple hetero-nuclear decoupling pulse sequence .**

We'll draw upon our "ideal" decoupling simulation (previous chapter) and simply replace the active Hamiltonian to include all scalar couplings as well as a term for the applied rf-field. The important part of this simulation is that it must be performed in the rotating frame of the applied rf-field. That is, in order for our Hamiltonian to remain constant we must be in a rotating frame with the applied field.

4.3 CWdecGP0.cc

CWdecGP0.cc

```
/* CWdecGP0.cc ***************************************************************-*-c++-*-
**                                                                             **
**  This program calculates a 1D spectrum during CW decoupling.  It is         **
**  quite simple in that it first applies a hard 90 pulse to the               **
**  detection channel and no relaxation effects are considered during          **
**  the acquisition.                                                           **
**                                                                             **
**  Assuming a.out is the executable, you should be able to obtain a           **
**  13C decoupled proton spectrum using the command                            **
**                                                                             **
**            a.out CWdecGP0.sys 1024 1H 13C 5000 1.0                          **
**                                                                             **
**  then answering the question about the Nyquist frequency with a "u".        **
**                                                                             **
** Author:    S.A. Smith                                                       **
***                                                                           **
*********************************************************************************/

#include <gamma.h>

main (int argc, char* argv[])
 {
 cout << "\n\tGAMMA Decoupling Simulation: Gnuplot, No Relaxation\n";
//                              Read in Parameters
 int qn = 1;                                    // Parameter query number
 spin_system sys;                               // A spin system
sys.ask_read(argc,argv,qn++);                   // Read in/Ask for system
 cout << sys;                                   // Have a look at the system
 int t2pts;                                     // Block size
 query_parameter(argc, argv, qn++,              // Get number FID of points
          "\n\tAcquisition Size? ", t2pts);
 String IsoD, IsoCW;                            // Pulse/Detect, Decouple
 if(sys.homonuclear())                          // Set pulse/detect and
  {                                             // decoupling channels the
  IsoD = sys.symbol(0);                         // same if homonuclear
  IsoCW = IsoD;                                 // system input
  }
 else                                           // For heteronuclear systems
  {                                             // Ask for the channel types
  query_parameter(argc, argv, qn++,             // e.g. 1H, 13C, 19F, ...
     "\n\tPulse/Detect Channel? ", IsoD);
  query_parameter(argc, argv, qn++,
       "\n\tDecoupling Channel? ", IsoCW);
  }
 double gamB1;
 query_parameter(argc, argv, qn++,              // Get the decoupling strength
       "\n\tDecoupling Field Strength[Hz]? ", gamB1);

 double lwhh = 3.0;                             // Half-height linewidth
 query_parameter(argc, argv, qn++,              // Ask for apodization strength
          "\n\tApodization (Hz)? ", lwhh);
//                              Set Variables
 gen_op H = Ho(sys);                            // Set isotropic Hamiltonian
 gen_op Heff = H + gamB1*Fx(sys,IsoCW);         // Set the effective Hamiltonian
 gen_op detect = Fm(sys, IsoD);                 // Set detection operator to F-
 block_1D fid(t2pts);                           // A block for acquisition
//                          Set Up Spectral Parameters
 double NyqF = query_Nyquist(sys, IsoD);        // Choose a Nyquist frequency
 double dt = 1.0/(2.0*NyqF);                    // Dwell time, quadrature
 double SW = 2.0*NyqF;                          // Total Spectal width +/- Nyquist
//                      Implement Pulse - Acquisition Sequence
 gen_op sigma0 = sigma_eq(sys);                 // Start at equilibrium
 gen_op sigma1 = Iypuls(sys, sigma0, 90);       // Apply (PI/2)y ideal pulse
 FID(sigma1,detect,Heff,dt,t2pts,fid);          // Calculate FID under Heff
//                          Process and Output Data
 double RR = (lwhh/2)*HZ2RAD;                   // Set apodization rate
 double tt = double(t2pts-1)*dt;                // Total FID length
 row_vector vex=Exponential(t2pts,tt,0.0,RR,0); // Block for apodization
 row_vector  fidap = product(fid,vex);          // Apodized the FID
 row_vector data = FFT(fidap);                  // Apply FFT
 cout << "\n\n";                               // Keep screen nice
 cout.flush();                                  // Insure writing all done
 GP_1D("data.gnu", data, 0, -(SW/2), (SW/2));   // Output gnuplot ASCII file
 ofstream gnuload("gnu.dat");                   // File of gnuplot commands
 gnuload << "set data style line\n";            // Set 1D plots to use lines
 gnuload << "set xlabel \"W2(Hz)\"\n";          // Set X axis label
 gnuload << "set ylabel \"Intensity\"\n";       // Set Y axis label
 gnuload << "set title\"Spectra\"\n";           // Set plot title
 gnuload << "plot \"data.gnu\"\n";              // Plot FIDs in gnuplot
 gnuload << "pause -1 \'<Return> To Exit \n";   // Pause before exit
 gnuload << "exit\n";                           // Now exit gnuplot
 gnuload.close();                               // Close gnuplot command file
 system("gnuplot \"gnu.dat\"\n");               // Plot to screen
 cout << "\n\n";                               // Keep the screen nice
 }
```

How does this account for the rf-field rotating frame? It actually works in multiple rotating frames, one for each isotope type. Only the decoupler channel is in the rotating frame of the field, the rest have all of their assocated spins referenced to some other frame. Note that this is an approximation! It assumes that there are no re-

sidual effects from working in such frame and throws away the corresponding time dependent Hamiltonian terms[1].

4.4 Homonuclear Decoupling During Acquisitions

For our first example we'll begin with a very simple simulation, a decoupled NMR spectrum.

Homonuclear Decoupling Sequence

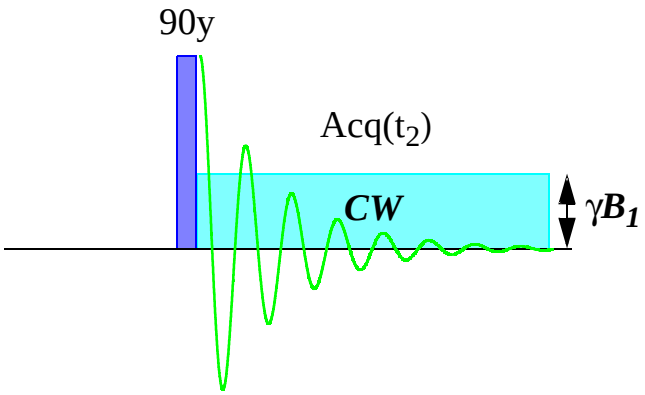


Figure 19-9 A simple decoupling pulse sequence .

1. In isotropic systems the active Hamiltonian is taken as a combination of Zeeman and scalar coupling terms. It is the latter which becomes time-dependent when switched into a multiple rotating frame. However, if the rotating frames are far apart such terms are negligible. This will almost always be the case since the rotating frames are separated by the isotope Larmor frequency differences. Exceptions will be when the Bo field is turned down and/or when working with heteronuclei which happen tohave similar Larmor frequencies. In that rare event one is forced to work in a single rotating frame (but that isn't really a problem then anyway). A much worse situation occurs when one attempts to work in a multiple rotating frame on the same channel because the time-dependent terms are then NOT small.

4.5 CW Decoupling Profile

Now we'll generate a profile. Such experiments are usually done in order to evaluate decoupling performance, typically for some multiple pulse decoupling scheme. We'll do one for CW decoupling so it won't be anything new when encountered in some broad-band decoupling simulation later in this book. The pulse sequence we will implement is the same one we've used before, shown in the following figure.

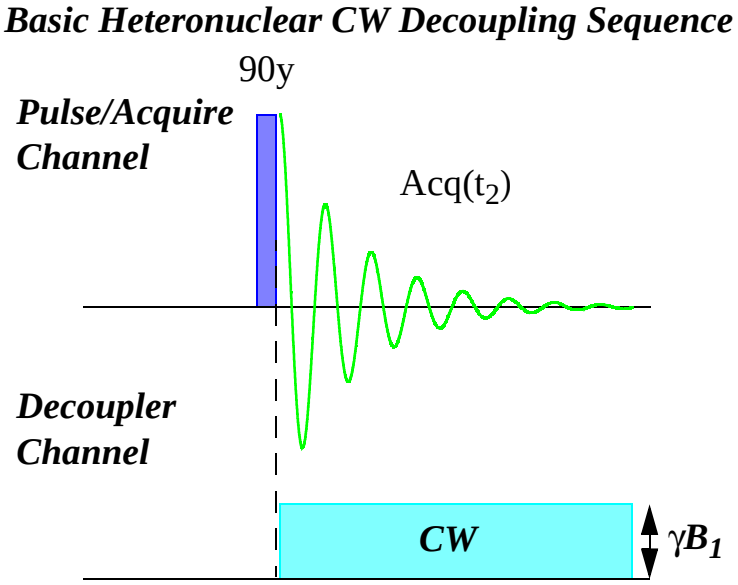


Figure 19-10 **A simple hetero-nuclear decoupling pulse sequence .**

The experiment is normally performed on a 2-spin heteronuclear spin system. A typical sample (ala Freeman and Shaka) would be ^{13}C labeled formate where decoupling is applied on the carbon channel and the aldehydic proton is detected. The decoupler rf offset is changed with each repeated experiment and the proton spectra from these placed side by side.

First, let us pick a simple spin system. The file below, named CWdecprof0.sys, will be used as input. You can't get any simpler than this.

CWdecprof.sys

```
SysName   (2) : CHprof   - Name of the Spin System
NSpins    (0) : 2        - Number of Spins in the System
Iso(0)    (2) : 13C      - Spin Isotope Type
Iso(1)    (2) : 1H       - Spin Isotope Type
v(0)      (1) : 0.0      - Chemical Shifts in Hz
v(1)      (1) : 0.0      - Chemical Shifts in Hz
J(0,1)    (1) : 166      - Coupling Constants in Hz
Omega     (1) : 500      - Spect. Freq. in MHz (1H)
```

Note that both the carbon and proton are set to be on resonance. For the proton this means that our generated spectra will be symmetric about 0 Hz. For carbon we want to start at 0 Hz so we can reference out decoupler rf offset. Since we are not including relaxation, and the scalar coupling is weak, the Bo field strength will have no influence on our results.

Now lets have a look at the code itself. We will modify one of the previous programs to ask the user for information pertaining to the profile then add a loop over the offset values. The only “new” aspect here is that the spectra are placed into a single block (row vector). All of this I/O and looping makes the program get a bit long....

CWdecGP0.cc

```
/* CWdecprof0.cc ***************************************************-*-c++-*-
**                                                                       **
**          GAMMA CW Decoupling Profile Simulation Program               **
**                                                                       **
** This program simulates CW decoupling.  It produces a decoupling       **
** profile, i.e. a series of 1D plots stacked side to side where each    **
** plot is produced at a different decoupler offset. The user specifies  **
** the number of offsets and the offset increment.                       **
**                                                                       **
** The program is designed for a 2-spin heteronuclear system.  The user  **
** chooses the spin system and which channel to detect.  The pulse       **
** channel is set the same as the detection channel and the decoupler    **
** channel set to the hetero-nucleus.                                    **
**                                                                       **
** No relaxation is accounted for, the user must set a line-broadening   **
** for the acquired spectra.                                             **
**                                                                       **
**  Assuming a.out is the executable, you should be able to obtain a     **
**  13C proton decouped profile using the command                        **
**                                                                       **
**          a.out CWdecprof0.sys 1H 2000 1000 10 1024 100 5              **
**                                                                       **
** Author:    S.A. Smith                                                 **
** Date:      5/9/1996                                                   **
** Last Date: 4/17/98                                                    **
** Copyright: S.A. Smith, September 1995                                 **
** Limits:    1.) Needs >= GAMMA 3.5                                     **
**            2.) Output is gnuplot interactive.                         **
**            3.) Assumes an isotropic spin system.                      **
**            4.) Relaxation effects are included.                       **
**                                                                       **
***************************************************************************/

#include <gamma.h>                                  // Include GAMMA

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA CW Decoupling Profile Simulation\n";
//                              Read in Parameters
 int qn=1;
 spin_system sys;                                   // Declare dynamic system sys
 sys.ask_read(argc,argv,qn++);                      // Ask for file name of sys
 if(sys.spins() !=2 || sys.homonuclear())           // Insure proper input system
  {
  cout << "\n\tSorry, this program is designed to take a two"
       << " spin heteronuclear spin system ONLY!\n\n";
  exit(-1);
  }
 String IsoD, IsoCW;                                // Detector, Decoupler channel
 query_parameter(argc, argv, qn++,
   "\n\tDetection Channel (e.g. 1H, 13C, ...)? ", IsoD);
 if(sys.symbol(0) == IsoD)      IsoCW = sys.symbol(1);
 else if(sys.symbol(1) == IsoD) IsoCW = sys.symbol(0);
 else
  {
  cout << "\n\tSorry, there are no spins of type " << IsoD
       << " present.  Try another detection channel!\n\n";
  exit(-1);
  }
//                          Set Decoupling Profile Parameters
 double gamB1;
 query_parameter(argc, argv, qn++,                  // Get the decoupling strength
  "\n\tDecoupling Field Strength[Hz]? ", gamB1);
 double SWprof;
 query_parameter(argc, argv, qn++,                  // Get the profile range
  "\n\tTotal Profile Spectral Width[Hz]? ", SWprof);
 int NO = 30;                                       // # Of Offsets (on each side)
 query_parameter(argc, argv, qn++,                  // Get # offsets
  "\n\tNumber of Positive Decoupler Offsets? ",
                                  NO);
 double offset = SWprof/(2.0*double(NO));           // Set offset increment (Hz)
 double totaloff = -SWprof/2.0;                     // Starting offset value (Hz)
//                        Set Individual Spectrum Parameters
 int t2pts;
 query_parameter(argc, argv, qn++,                  // Get # offsets
  "\n\n\tNumber of Points Per Offset? ", t2pts);
 double SW;                                         // Nyquist frequency
 query_parameter(argc, argv, qn++,                  // Get # offsets
  "\n\n\tSpectral Width Per Offset? ", SW);
 double lwhh = 3.0;                                 // Half-height linewidth
 query_parameter(argc, argv, qn++,                  // Ask for apodization strength
          "\n\tApodization (Hz)? ", lwhh);
//                              Set Variables
 double dt = 1.0/(SW);                              // Dwell time, quadrature
 double RR = (lwhh/2)*HZ2RAD;                       // Set apodization rate
 double tt = double(t2pts-1)*dt;                    // Total FID length per offset
 row_vector vex=Exponential(t2pts,tt,0.0,RR,0);     // Block for apodization
//                          Output Specified Parameters
```

```
cout << "\n\tSet FID Dwell Time To \t" << dt;
cout << "\n\tSet Spectral Width To \t" << SW;
cout << "\n\tSet LW @ Half-Height To \t" << lwhh;
cout << "\n\tProfile Entry Points  \t" << t2pts;
cout << "\n\tProfile Total Points  \t" << t2pts*(2*NO+1);
cout << "\n";
cout.flush();
sys.offsetShifts(-NO*offset, IsoCW);
block_1D bdata(t2pts);                          // Block for spectrum
row_vector fidap;                               // For apodized FID
row_vector data(t2pts);                         // Block for spectrum
row_vector profile((2*NO+1)*t2pts, complex0);   // Block for profile
int K=0;
String nam;
String st="cwoff";
String fi=".asc";
double actoff = totaloff;                       // Current offset value (Hz)
int len = 2;
if(NO >9)   len++;
if(NO >99)  len++;
if(NO >999) len++;
String iform = String("%") + dec(len) + String("i");
for(int ov=-NO; ov<=NO; ov++)
 {
 cout << "\n\tSimulating Offset " << form(iform, ov)
      << " at " << form("%8.2f", actoff) << " Hz";
 cout.flush();
 gen_op H = Ho(sys);                            // Set isotropic Hamiltonian
 gen_op Heff = H + gamB1*Fx(sys,IsoCW);         // Set the effective Hamiltonian
 gen_op detect = Fm(sys, IsoD);                 // Set detection operator to F-
 gen_op sigma0 = sigma_eq(sys);                 // Set density mx equilibrium
 gen_op sigmap=lypuls(sys,sigma0,IsoD,90.0);    // Apply a 90 pulse
 FID(sigmap,detect,Heff,dt,t2pts,bdata);        // Calculate FID under Heff
 data = bdata;                                  // Make it a row_vector
 fidap = product(data,vex);                     // Apodized FID this offset
 data = FFT(fidap);                             // From propagator time domain
 for(int k=0; k<t2pts; k++, K++)                // Store spectrum in profile
   profile.put(data.get(k), K);
 sys.offsetShifts(offset, IsoCW);               // Offset detection channel
 actoff += offset;                              // Track current offset value
 }

double Hzppt = SWprof/(2.0*double(NO));
double sumoff = Hzppt*double(2*NO+1)/2.0;
GP_1D("profile.gnu",profile,0,sumoff,-sumoff);  // Output spectrum in gnuplot
cout << "\n\n";                                 // Clean up screen
cout.flush();                                   // Flush output before gnuplot
ofstream gnuload("CW.gnu");                     // File of gnuplot commands
gnuload << "set data style line\n";             // Set 1D plots to use lines
gnuload << "set xlabel \"Offset(Hz)\"\n";       // Set X-axis label
gnuload << "plot \"profile.gnu\"\n";            // Plot the spectrum
gnuload << "pause -1 \'<Return> To Exit \n";    // Pause before quitting gnuplot
gnuload << "exit\n";                            // Exit gnuplot
gnuload.close();                                // Close gnuplot command file
system("gnuplot \"CW.gnu\"\n");                 // Plot to screen
FM_1D("CWprof.mif",profile,14,14,sumoff,-sumoff); // FM 1D plot file
cout << "\n\n";                                 // Keep the screen nice
}
```

Now we can run the program. Assuming that we have compiled the program to produce an executable "a.out" (this will likely be a.exe on a Windoze system), we can run the program with the command

a.out CWdecprof0.sys 1H 2000 1000 10 1024 100 5

But one can also run the program interactively. Here is the dialog associated with such a run.

```
|gamma1>a.out

GAMMA CW Decoupling Profile Simulation
Spin system filename? CWdecprof0.sys
Detection Channel (e.g. 1H, 13C, ...)? 1H
Decoupling Field Strength[Hz]? 2000
Total Profile Spectral Width[Hz]? 1000
Number of Positive Decoupler Offsets? 10
Number of Points Per Offset? 1024
Spectral Width Per Offset? 200
Apodization (Hz)? 5

Set FID Dwell Time To          0.005
Set Spectral Width To          200
Set LW @ Half-Height To        5
Profile Entry Points           1024
Profile Total Points           21504

Simulating Offset -10 at  -500.00 Hz
Simulating Offset  -9 at  -450.00 Hz
Simulating Offset  -8 at  -400.00 Hz
Simulating Offset  -7 at  -350.00 Hz
```

```
Simulating Offset  -6 at  -300.00 Hz
Simulating Offset  -5 at  -250.00 Hz
Simulating Offset  -4 at  -200.00 Hz
Simulating Offset  -3 at  -150.00 Hz
Simulating Offset  -2 at  -100.00 Hz
Simulating Offset  -1 at   -50.00 Hz
Simulating Offset   0 at     0.00 Hz
Simulating Offset   1 at    50.00 Hz
Simulating Offset   2 at   100.00 Hz
Simulating Offset   3 at   150.00 Hz
Simulating Offset   4 at   200.00 Hz
Simulating Offset   5 at   250.00 Hz
Simulating Offset   6 at   300.00 Hz
Simulating Offset   7 at   350.00 Hz
Simulating Offset   8 at   400.00 Hz
Simulating Offset   9 at   450.00 Hz
Simulating Offset  10 at   500.00 Hz
```

The profile produced from this is shown in the figure below.

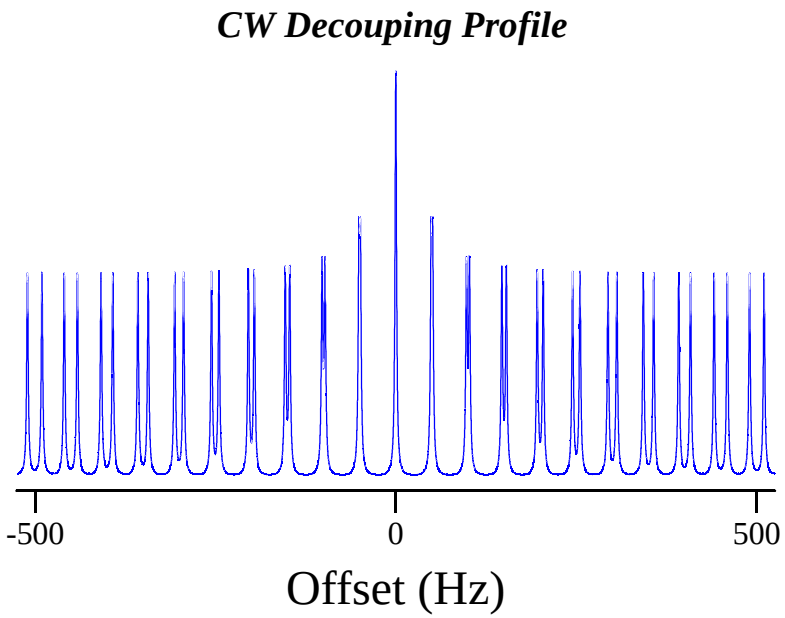


Figure 19-11 **CW decoupling profile genetared from CWdecprof0.cc.**