FMP-Pulse Program Library and Setup tools

I hope that these pulse program and python libraries are easy to read, use, and maintain, as well as relatively complete.

In compiling this pulse sequence library, I made several stylistic choices that are consequent for working with biological (i.e. protein) samples in a lab with NMR consoles of various ages. I have ignored nuclei that are not ^{1}H, ^{15}N, ^{13}C, or ^{2}H. Pulses are currently defined by nucleus, instead of spectrometer channel, to ease bookkeeping across multiple acquisition modes and pulse sequences. To ease maintenance, protection circuits and multiple-pulse recoupling schemes are compiled into external files, so that changes in a mixing scheme or protection will automatically propagate throughout the library. Finally, many experiments allow the user to customize their experiment using flags, for example applying constant duty cycle experiments or compound 180° pulses.

MAS NMR is typically more complex than its solution counterpart. However, MAS dependent match conditions can be calculated with reasonable precision provided the probe response and MAS frequency are known. A large number of python programs are provided to calculate power levels for several pulse sequence elements that are useful in protein ssNMR.

I would encourage everyone to report any bugs or double use of parameters (some double use may be intentional) I would be glad to include your favorite experiments if you'll send in an example. I sometimes misinterpret the literature, so if you could send in a description of your set-up or your calculations, it will help me write the setup script.

External Files:

External files containing repetitive or complicated sequence elements make maintaining the pulse sequence library easier, but make the pulse sequence itself less transparent. These external files can be accessed with the following commands:

edcpulall: Open an editor window for every file called in the pulse program
edcincl: Open each file with a "*.incl*" extension
edcsubr: Open all "*.subr*" files
edcprot: Open all protection files

Includes files (extension "*.incl*") contain channel definitions, rotor period calculations, and protections.

Subroutine files (extension ".subr") contain timing calculations needed to properly run the experiment, subroutines used in the body of the pulse sequence, and any phases and parameter definitions needed in the main sequence. It is necessary to copy and paste the parameter and phase definitions into the main sequence or the program will not compile.

Pulse Sequence Framework and Usage:

Style/Format

Each pulse sequence opens with a brief citation, usage and optimization requirements, and topspin keywords. The Topspin keywords allow the user to quickly filter the files into appropriate categories. The channels and mixing are then defined via external file calls. Initial evolution times and other protected times are then calculated. There is then a preparation period that is only run at the beginning of the pulse sequence. This section contains most of the protection files. The nested loops of the pulse sequence begin with "Start" (or in some cases "2") and end at HaltAcqu. The phase lists and parameter lists follow.

Sequences have been tested on Avance I and II spectrometers and work for the older spectrometers. All have been forward compatible for me, but perhaps not optimal due to faster time events. The need for back compatibility is the reason for several parameter naming decisions, such as using fast shaped pulses and power levels instead of "sp#" as the power level (`spf# pl#` *vs* `sp#`). Avance and Topspin 3 versions of the code are currently underway.

Spectrometer timing delays are coded with "hard" times (i.e. "`0.3u do:H`"). These delays are spectrometer dependent, so it may be useful to you to update with shorter timings. My motivation was to have a single set of experiments that worked on all consoles (some quite old). The hard coded timings may, however, be too short for some older spectrometers. Usually, changing a troublesome delay to 1u is sufficient for the program to execute. It is good practice to confirm the spectrometer output with an oscilloscope if you suspect something is not as it should be.

Naming Pulse Sequences

Each experiment is named according to its polarization transfer pathway, the type of mixing, and then any special features. Any nucleus/dimension that has a time evolution (i.e. can be frequency labeled) is capitalized. Nuclei in lower-case letters are not evolved. The final capitalized nucleus is the observe nucleus. 'R' denotes a relaxation dimension (using the "vd" convention). All polarization transfers are by CP, unless specified.

For example, an "hCC" experiment is a ^{13}C-^{13}C two-dimensional carbon-detected experiment where hCH is a ^{13}C-^{1}H two-dimensional proton-detected experiment.

Curly braces "{ }" indicate a recoupling period
hC{HC}C is a 3D experiment with an HC recoupling scheme after mixing
h{HC}CC is a 3D experiment with recoupling scheme before t1 evolution

Square braces "[ ]" indicate a simultaneous chemical shift evolution
h[NC]H is a 2D HSQC where both N and C are excited and evolved.

Dimensions

Unfortunately, the code as currently written must be run from its highest dimension. You can take a 1D by setting all other dimensions to take 1 point and then changing the processing to 1D processing. There is no evolution time when only 1 point is taken.

Example: To change from fmp.hC to fmp.hCC_DARR, but still only acquire a 1D experiment, change the pulse program name, change the acquisition parameters to 2D, change the number of increments in F1 to 1, and the change the processing parameters to 1D. (using MrAcq should put reasonable acquisition parameters and set things to a 1D)

The initial evolution time point is pre-calculated to give (90, -180) phasing. Certain DQ and Inept transfer experiments will give (0,0) phasing which should be listed in the header of the experiment.

Zgoptns (Options)

In zgoptns turn on an option using –DXXXX

Options include but are not limited to the following:

CDC: Constant Duty Cycle (always decouple up to the maximum indirect dimension evolution time or cnstN where N=31-FX (FX=F1,F2,F3))
DP: Direct Polarization (replace CP with hard pulse)
S180: Use a soft 180° pulse instead of a hard 180°.
CPRE: Carbon Presaturation
PRE: Proton Presaturation
Purge: Carbon soft 90° pulse during longitudinal (DARR, PDSD, RFDR) period
cmpd: Use a compound (composite) 180° pulses (i.e. in RFDR)

Note: options only turn on with "-D" in front, so a Direct Polarization experiment is indicated with "-DDP"

Naming Pulses

The naming conventions for the FMP library are compiled in a separate document; please report or correct any inconsistencies. The python libraries provide a means to convert between conventions, which will be described below. An effort has been made to have continuity between pulse and power-level names, but this has proven to be difficult given the naming constraints in Bruker's pulse code.

Python Setup Scripts and Utilities

Included is a large set of programs to calculate many power levels and pulse widths used in MAS NMR. The hard pulses must be calibrated for these programs. The calculations generally do not give the optimal condition due to differences in the probe response between short and long pulses, and slight errors in the hard pulse calibrations. The error is usually less than 2 dB, so it is necessary to check the calculation with an optimization.

Running these scripts will pop-up several windows which ask you to provide and/or verify various conditions. You are free to change the suggested value; the program will use the numbers that you have entered to calculate its conditions regardless of their suitability or safety. When there is a missing file, such as a shaped pulse or compound pulse decoupler program, a dialog should open up to allow you to interact with topspin's file system. This interaction occasionally causes the program to exit due to an unfound file, however the next time the routine is run it will be successful.

Power calculations (CP)

HC, HN, NCA, NCO

CP is used throughout ssNMR experiments for heteronuclear polarization transfer. In all cases, each channel's hard pulse and its power define the maximum allowed B1 field. Because one or both of the channels utilize a shaped pulse, the calculation integrates the shaped pulse and corrects for the average power deposited. In the case of proton CP, the program attempts to use the highest B_1 field (matched to $(n\pm\frac{1}{2})*\omega_r$) that is still less than the maximum B_1 corresponding to that channel. To clarify, the channel with the lowest maximal B_1 field is used to determine a CP field that is $(n-\frac{1}{2})*\omega_r$ that is still below the maximum B_1 for that channel. The other channel (whose match condition is $(n+\frac{1}{2})*\omega_r$) is checked against its maximum B_1, and n is decreased by one until n=0 or both channels are below their thresholds. If n=0, a double quantum match condition is calculated using $\frac{3}{4}*\omega_r$ for ^1H and $\frac{1}{4}*\omega_r$ for the X channel. SPECIFIC CP is calculated to be $B_1(CA)=3/2\ \omega_r$ and $B_1(N)=5/2\ \omega_r$ for an NCA transfer, and $B_1(C')=7/2\ \omega_r$ and $B_1(N)=5/2\ \omega_r$ for an NC' transfer. The same power check is applied as in the HX calculations.

Power calculations (Decoupling)

HDec

This program will suggest 75 kHz decoupling and report back a power level and a 170° pulse length. The decoupling field can be modified, but the power level and pulse length are not alterable. Be aware that any decoupling sequence that doesn't use ~180° pulses must be recalculated.

N, C, and D versions are currently not implemented.

Soft Pulses (Carbon)

The soft-pulse's shape file is integrated, then the integration and the hard pulse and soft pulse lengths are used to determine the power needed during the soft pulse. There is no consideration given to the bandwidth of the pulse; the default pulse length is $1.5*\omega_r$.

Pulse Names and Library Definitions

The python scripts change their behavior with respect to the topspin version, and potentially to the user definitions found, for example in fmpTS3p2.py or bruTS2p1.py.

These library files define functions that are needed to read and write in different topspin versions. At the top of each file is a list of descriptive pulse names and their corresponding variable in the pulse sequence library. The user is invited to modify the definitions for use with their own pulse sequence library. This will allow them to use these python programs with a minimum amount of work.

These files **MUST** be named with your library prefix or suffix (i.e. fmp or wtf) and then "TSXpY.py" where X is the topspin release, and Y is the version (so Topspin 3.2 is TS3p2) Your prefix/suffix must be added to "GetLib.py" in the "PPvers" (line #3).

Merger and Migration (MrMerger and MrMigrate)

Experimental parameter sets can be merged/ filled using the command "MrMerger". For example, if there is an empty dataset, the parameters of the previously calibrated experiments can be mined for the empty dataset based on the pulse program name. See MrMerger for options.

Parameters can be migrated between libraries using MrMigrate. This is a silent migration; any repeated pulse sequence elements may be overwritten. (For example, if NCA or NCO use different pulse names in one library, but use the same name in the other library, one or the other will be over-written)

MrAcq

Acquisition parameters are calculated with respect to the MAS rate and pre-defined sweep-widths (in ppm) according to the pulse sequence name and dimension. (See MrAcq for detailed options)

MrMAS

Recalculate match conditions for a new MAS rate using previously entered conditions. The CP conditions are checked for exceeding maximum B_1 fields.

Miscellaneous Mixing

There are several additional python scripts that will calculate power levels needed for homonuclear recoupling. Symmetry based pulse sequences ($SPC5_2$, $SPC5_3$, POSTC7) use straightforward calculations. The DREAM module offers several options to find the match condition depending on the type of carbon. Various types of TOBSY, and EXPORT, DUO, R^2T, and others are usable, although as yet the pulse sequences have not been moved into "circulation".

Installation:

It is recommended to install the pulse programs in a separate library. This keeps the standard and user libraries intact, and allows you to quickly move to these pulse programs.

This is the way I installed the programs at FMP:

Create a folder in the 'pp' folder, for example:

```
>mkdir /opt/topspin/exp/stan/nmr/list/pp/ppfmp
```

put the pulse programs into this folder and create a link.

```
>cp –r folderpath /opt/topspin/exp/stan/nmr/list/pp/ppfmp
>ln –s /opt/topspin/exp/stan/nmr/list/pp/ppfmp homes/guest/ppfmp
```

Similarly for the python scripts:

```
>mkdir /opt/topspin/exp/stan/nmr/py/BioPY
>cp –r folderpath /opt/topspin/exp/stan/nmr/py/bioPY
>ln –s /opt/topspin/exp/stan/nmr/py/BioPY homes/guest/pyfmp
```

In Topspin, open the change pulse program window. Under Options, choose "Manage Sources". Enter the path for the symbolic link in the pulse programs window, in this case "/homes/guest/ppfmp" into the pulse program window. You also need "/homes/guest/ppfmp/INCL_SUBR" Similarly, enter the path to the link for the python programs into the python window. In this case, it is "/homes/guest/pyfmp". Restart TOPSPIN.

Note: Topspin does not allow you to point to the /opt directory tree. We back up the /opt trees regularly at the FMP, but not the /homes/guest tree. I wanted use these backups but it is not a requirement to do it like this. You can simply put the folders in a convenient place, (as long as it's not in the /opt tree) and work from there.

After you get Topspin up again, you'll be able to pull down to the appropriate folders for the pulse programs and python scripts when you run edpul and edpy.

The python scripts may not immediately run because the paths to the modules are hard coded. The release will assume that the files are installed in "*topspinhome*/exp/stan/nmr/py/BioPY" (where *topspinhome* is */opt/topspinX.Y/*)There are two options for this to work. My preferred option is to supply symbolic links to the actual code. Otherwise you can add the destination to topspin's lookup table as described on pages 3 and 4 of the Topspin "Python Programming" manual.

If you identify an inconsistency, have a suggestion for improvements, or especially if you have a contribution please contact me (Trent Franks, franks@fmp-berlin.de) and I'll see about including it.

Commands

Manipulate Multiple Parameters
(stuffing and merging)

MrMAS	MrSetup	MrMerger	MrMigrate	MrAcq

CP

HC	NCA	ChhC
HN	NCO	NhhC

Soft Pulses

CApurge	pS6	PurgeS6	CTUC
COpurge	pS7	PurgeS7	
CArefocus	pS8	InvS8	
COrefocus	pS9	InvS9	

Mixing, NOT CP

TOBSY	POST_C7	DUO	R2Twave	SPC53
C931	C7	EXPORT	SPC5	
C961	DREAM	R2T	SPC52	

Decoupling

HDec.py	Dec.py

Editing

edcincl	edcsubr	edcprot	edcpulall

Make various files
(variable delay lists, cpd files, wave files)

mkT2list	mkT2vd	TAN	MkSPINAL

Multiple parameter Manipulation

MrMAS: Calculate new match conditions for a new MAS rate from the parameters themselves (not from the hard pulses)

MrSetup: Use hard pulses to calculate many match conditions

MrMerger: Put parameters from other experiments into the current experiment (EXPNO -1 is default)

MrMigrate: Translate parameter definitions from one dataset into another. There must be a file that defines the parameter defaults.

MrAcq: Stuff acquisition parameters into the acq file.

CP

HC & HN: Calculate H-X CP match conditions from hard pulses. The calculation finds the highest B1 fields allowed by probe performance, then sets a match condition where the B1 fields are $\omega_r*(n\pm1/2)$. The CP pulse shapes are integrated to account for the change in amplitude. If no CP match condition is found, the routine defaults to $\omega_1(^1H)=3/4*\omega_r$; $\omega_1(X)=1/4*\omega_r$

NCA: Calculate N-C SPECIFIC CP conditions from hard pulses. The calculation starts from $\omega_1(^{15}N)=5/2*\omega_r$ $\omega_1(^{13}C)=3/2*\omega_r$. If the initial guess for the CP match condition can't be obtained, the condition is lowered by one ω_r until the program reverts to $\omega_1(^{15}N)=3/4*\omega_r$ $\omega_1(^{13}C)=1/4*\omega_r$.

NCO: Same as NCA but with initial values $\omega_1(^{15}N)=5/2*\omega_r$ $\omega_1(^{13}C)=7/2*\omega_r$.

ChhC & NhhC: Propagate HC and HN match conditions into the appropriate parameters.

CTUC: Calculate all soft pulses and set delays for the CTUC experiment.

Soft Pulses

All: Calculate pulse amplitude by integrating the shape and comparing to the hard pulse calibration. The pulse is assumed to be an excitation (6 & 7) or inversion (8 & 9) pulse. The offset is then placed for the appropriate nucleus (Ca :6 & 8; C': 7 & 9).

Mixing

TOBSY: Interact to calculate conditions for C931 or C961 or the adiabatic pulse C542.

C931: Calculate $C9_3^1$ TOBSY homonuclear scalar-transfer condition. No warnings issued if excessive amplitude is calculated.

C961: Calculate $C9_6^1$ TOBSY homonuclear scalar-transfer condition. No warnings issued if excessive amplitude calculated.

C7 or POSTC7: Calculate $C7_2^1$ homonuclear dipolar-transfer condition. No warnings issued if excessive amplitude calculated

DREAM: Calculate a ^{13}C DREAM condition and pulse shape for various ^{13}C chemical shifts. The average chemical shift is accessed

MrMerger: Put parameters from other experiments into the current experiment (EXPNO -1 is default)

MrMigrate: Translate parameter definitions from one dataset into another. There must be a file that defines the parameter defaults.

MrAcq: Stuff acquisition parameters into the acq file.