Cambridge CS

Compressed sensing reconstruction for multidimensional NMR spectroscopy

EMBO NMR Workshop 2019
'P24 - Non-uniform sampling in NMR'

Biomolecular NMR Zentrum, TU München

Dr. Mark J. Bostock*
Helmholtz Zentrum München & BNMRZ TU München

Dr. Daniel Nietlispach†
Department of Biochemistry, University of Cambridge, UK

26th July – 2nd August 2019

*mark.bostock@helmholtz-muenchen.de
†dn206@cam.ac.uk

Contents

1 Overview

Methods for improving the time-efficiency of NMR data acquisition have been popular since the early days of multidimensional NMR [1]. Such methods have focussed around recording a subset of the full data matrix (non-uniform sampling, NUS) and are coupled with a variety of processing methods due to the restriction of the Fourier transform to using regularly sampled data [1–9]. NUS approaches are typically used to improve resolution, sensitivity per unit time, or reduce the recording time for spectra, or a combination of all three [6, 10]. Compressed sensing (CS) was pioneered in the field of information theory [11–13] and quickly found applications in a number of fields including MRI [14] and more recently NMR as a new method for processing NUS data [15–18]. CS has been demonstrated to be a powerful method to reconstruct undersampled spectra with high fidelity, even for weak peaks in spectra with a high dynamic range [18]. Consequently CS has rapidly gained popularity in NMR spectroscopy. A number of reconstruction packages for NUS data have been discussed during this course. The aim of the Cambridge CS package is to provide an easy-to-use interface and scripting approach for reconstruction of NUS data using a variety of available CS algorithms. The aim of this practical is to introduce you to using the Cambridge CS package and to give you an overview of how NUS data can be successfully acquired and reconstructed. This practical handout will hopefully give you a good overview of how to implement CS reconstructions of NUS data in your own labs. Further details on accessing the software are included at the end.

2 Launching Cambridge CS

Cambridge CS is installed on the BNMRZ computer system.

Launch Cambridge CS with the following commands:

```
cambridgecs [OPTION] [Script file]
```

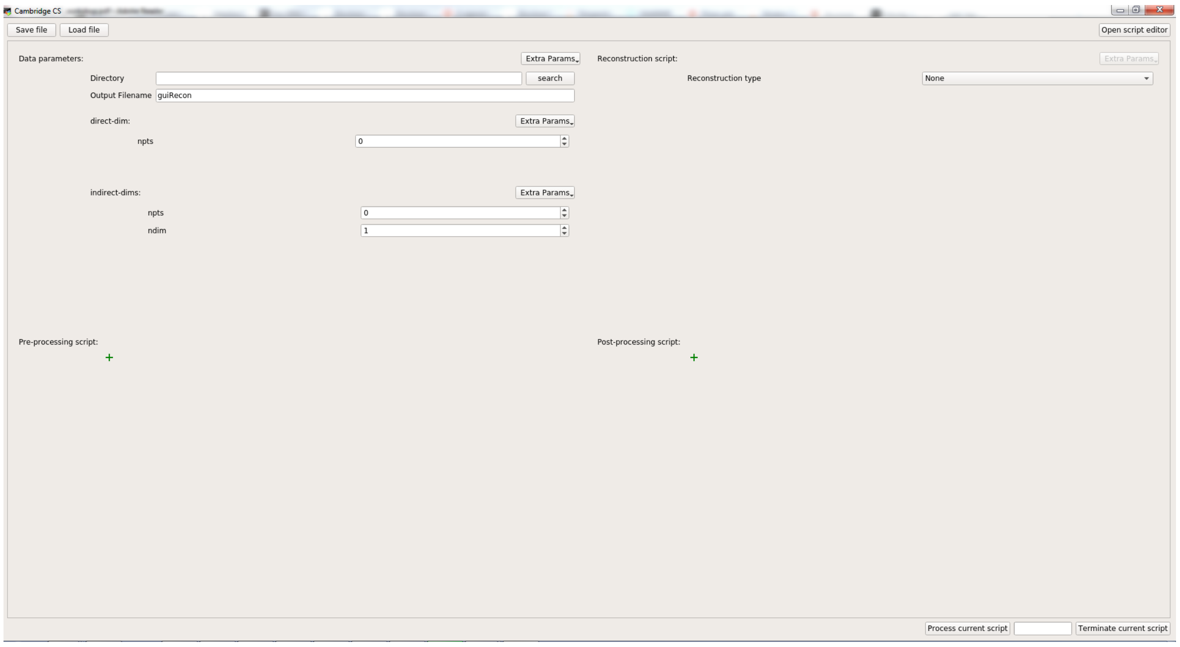
```
-a --aut
 Run in automatic mode. Does not launch GUI and requirements for user input are
 suppressed. Requires a script file.
-h --help
 Show help text
-g --gui
 Launch the GUI. The GUI may be launched without a script file in which case a
 script can be built within the GUI. Alternatively an existing script file may
 be loaded into the GUI and amended or run.
```

Here we will use

```
cambridgecs -g
```

to launch the GUI which should look as shown in Figure 1.

The GUI is divided into four quadrants: data parameters, pre-processing script, reconstruction script and post-processing script. At the top of the window are buttons to save or load a script (left-hand side). On the right-hand side is a button entitled ‘Open script editor’. The GUI is used to build a processing script. During the process, this may be viewed by clicking ‘Open script editor’, which brings up a text window containing the script commands. The



Directory
Output Filename guiRecon
npts 0
npts 0
ndim 1
Reconstruction type None

Figure 1: Cambridge CS GUI

script editor will be updated as commands are added or removed from the GUI. Alternatively commands may be typed directly into the script editor and the button 'Sync text' used to sync the editor window with the GUI. In the script window, '#' represents a comment line. At the bottom of the GUI window is the button 'Process current script'. This uses the commands setup in the GUI to run the processing script. 'Terminate current script' can be used to stop a reconstruction. The GUI window may be closed using the keys 'q' or 'Esc'.

3 Building a script

In the following section, we provide an overview of the various elements that make up the Cambridge CS GUI/scripts. We will then move on to attempting to process some NUS data.

3.1 Data parameters

This section contains information about the data set e.g. location, number of points, data type, sampling lists etc. The main parameters are displayed; further parameters which may be needed in some cases are found in the 'Extra Params' drop-down menus. At this stage, it is assumed that the data is a pseudo-2D dataset, regardless of the actual dimensionality. 'Direct-dim' refers to the directly observed acquisition dimension, whilst 'indirect-dims' refers to the remaining $n-1$ dimensions.

3.2 Pre-processing script

Continuing to treat the data as a pseudo-2D, this section is used to carry out FT processing of the acquisition dimension, which should be fully sampled in all cases. The commands used are based on Azara-style commands (Wayne Boucher, unpublished). See the Cambridge CS instruction manual and `http://cambridge2000.com/azara/` for more information on the commands.

Initially a green-cross is displayed. Clicking this brings up a list of commands, which can be selected. Once a command is entered, a red cross appears next to the command with green crosses above or below. The red cross is used to remove a commmand, whilst the green crosses allow a new command to be added above or below.

3.3 Reconstruction script

The reconstruction script section contains processing commands and information to reconstruct the indirect dimensions (typically undersampled). Once a reconstruction type is chosen (FT, or other CS alogrithms), processing parameters appear for the indirect dimensions, allowing phasing, zerofilling, window functions etc. to be added. The number of dimensions which appear is controlled by the value of 'ndim' under 'Data parameters'. As for the other sections, further parameters are available under the 'Extra Params' drop-down menus.

3.4 Post-processing script

The post-processing script is used to add additional post-processing commands once the reconstruction has been carried out. This section uses Azara (`http://cambridge2000.com/azara/`) which is bundled with this software. As for the 'Pre-processing script', a green cross allows access to a series of commands; in the Azara style each dimension is defined by 'script_com n' where 'n' is the dimension. In the Azara convention (which we also use for Cambridge CS), the acquisition dimension is $n = 1$ and the indirect dimensions are $n = 2, 3....$ Once a dimension is selected, another green cross provides access to the available commands. The input to the processing script is defined by 'Input Data', which may be time-domain ('Input Data > useTimeData') or frequency domain data ('Input Data > useFTData'), depending on the output from the 'Reconstruction script'.

3.5 Integration with NMRPipe

Cambridge CS is a stand-alone data processing package and full data processing of FT and NUS datasets can be carried out with the software and bundled Azara package. Cambridge CS can also be used within NMRPipe scripts. This will be discussed in more detail later in the handout. In this case, NMRPipe takes the place of the 'Pre-processing' and 'Post-processing' scripts. The 'Data parameters' and 'Reconstruction script' sections are then the only parts of the Cambridge CS script, which need to be set up.

In the next section, we will go through full processing with the Cambridge CS package, so you are familiar with all the functionality. In section 4.3 we will discuss how to use this in the context of an NMRPipe script.

4 Bruker NUS processing

Test data can be found in the folder */ms/data/prog/CambridgeCS_dist/testData*.
Copy the *testData* folder to your directory so that you can process this:

```
cp -r /ms/data/prog/CambridgeCS_dist/testData DESTINATION
```

If you are unsure how to use the Linux command line, please ask one of the instructors.

4.1 2D processing from scratch

Experiment 102 in the subdirectory *2D* is a 2D HSQC of ubiquitin recorded with 30% sampling (i.e. 40 points out of 128). We will now set up a processing script for this spectrum:

Data parameters

- Select the directory (where output files will be created)
- Choose a prefix for the output filenames
- In the Bruker style, data is stored in a numbered directory. Typically we keep this separate from the processed data. Select this directory for the *Raw data directory* field.
- Choose the NUS list (not essential here since the NUS list has the default name 'vclist' so it should be found automatically).
- Input the number of points in the direct dimension (this is the total number of real and imaginary points in the acquisition dimension).
- Input the number of points in the indirect dimension (this is the total number of real and imaginary points for *all* indirect dimensions). *Remember at this stage the programme is treating the data as a pseudo-2D, so all the indirect dimensions are grouped together. The nuslist will be used to arrange the data into the correct shape based on the number of dimensions, ndim.*
- Check that the number of indirect dimensions (ndim) is set to 1.

Pre-processing script

Select a suitable set of processing commands for direct dimension processing. We use the following:

```
complex
brukerGroupPhase                  71.625
gaussian_sw                       20.0    1.0    10000.0
zerofill                          1
fft
phase                             103.0    40.0    1.0
reduce
range                             220    480
```

`brukerGroupPhase` removes the first-order phase error from Bruker data; for AV data, set `brukerGroupPhase` to -1. This will automatically find the GRPDLY parameter from the *acqus* file. For older DRX data (used here) a conversion is applied where $GRPDLY = 0.5 \times lookup/\text{DECIM}$. The *lookup* value can be found from the table in Appendix A. The correction may also be applied as a first order phase correction of magnitude $360 * \text{GRPDLY}$ after FT.

`reduce` is used to throw away the imaginary component of the acquisition dimension which is not needed in normal CS data processing.

`gaussian_sw` is a Gaussian window function. Other window functions are available e.g. `sinebell` and `sinebell2` (squared sinebell). Multiple window functions can also be used, one per line.

The script can be created either directly in the GUI or in the script editor. Try both approaches.

Reconstruction script

- Select a suitable algorithm e.g. FT for Fourier transform processing or one of the CS algorithms e.g. IHT, IST, L1 for CS-reconstruction. We recommend using the IHT algorithm for CS reconstruction. This is a modified version of the iterative hard thresholding algorithm.
- Since the experiment has one indirect dimension a set of options for 'dim 2' are displayed. Set the zerofilling (1 times) and phasing (180, -360).
- The data is acquired with States-TPPI quadrature detection in dim 2. For Bruker data every other complex $t1$ point has to be multiplied by -1. Select the `mask_ppmm` command to achieve this. This command is found under 'Extra Params > mask > mask_ppmm'. The net effect is to shift the spectrum by half a spectral width.
- Choose a window function e.g. `sinebell 90`. Window functions are also found under the 'Extra Params' option for a given dimension.

Post-processing script

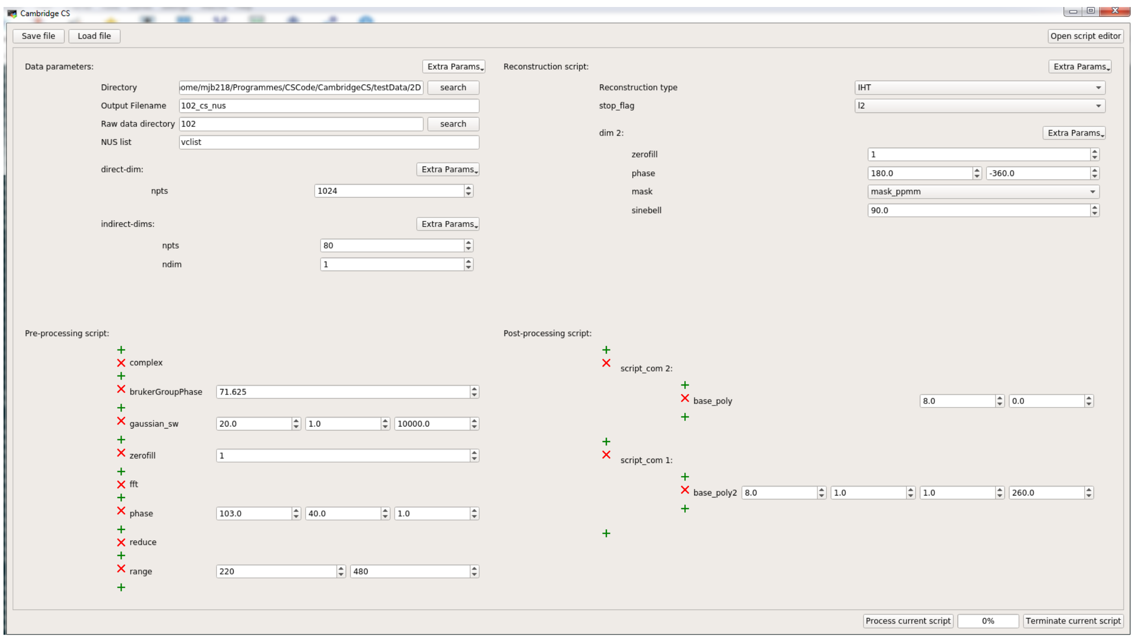
The default output from the 'Reconstruction script' is a frequency domain spectrum. Post-processing commands e.g. baseline correction or reversing dimensions may now be added. In this example we apply some simple baseline correction using the following script:

```
# Azara post processing instructions:
    post_processing_script:

        script_com 2:
            base_poly                 8.0    0.0
        end_script

        script_com 1:
            base_poly2                8.0    1.0    1.0    260.0
        end_script
    end_script
```

To implement these commands:

- Click the green cross and select `script_com 2`.
- Click the indented green cross and select 'Baseline correction' and then choose `base_poly`, and then the appropriate parameters.
- Click the unindented green cross below and select `script_com 1`.
- Click the indented green cross under `script_com 1` and choose 'Baseline correction > base_poly2' and input the appropriate parameters.

The final setup of the GUI should look similar to Figure 2. Meanwhile the accompanying script, built by the GUI, should look similar to Figure 3.



```
Data parameters:
    Directory            ome/mjb218/Programmes/CSCode/CambridgeCS/testData/2D
    Output Filename      102_cs_nus
    Raw data directory   102
    NUS list             vclist
    direct-dim:
        npts             1024
    indirect-dims:
        npts             80
        ndim             1

Reconstruction script:
    Reconstruction type  IHT
    stop_flag            l2
    dim 2:
        zerofill         1
        phase            180.0   -360.0
        mask             mask_ppmm
        sinebell         90.0

Pre-processing script:
    complex
    brukerGroupPhase     71.625
    gaussian_sw          20.0   1.0   10000.0
    zerofill             1
    fft
    phase                103.0  40.0  1.0
    reduce
    range                220    480

Post-processing script:
    script_com 2:
        base_poly        8.0    0.0
    script_com 1:
        base_poly2       8.0    1.0   1.0   260.0
```

Figure 2: Cambridge CS GUI setup for processing the 2D HSQC, experiment number 102, described in section 4.1.

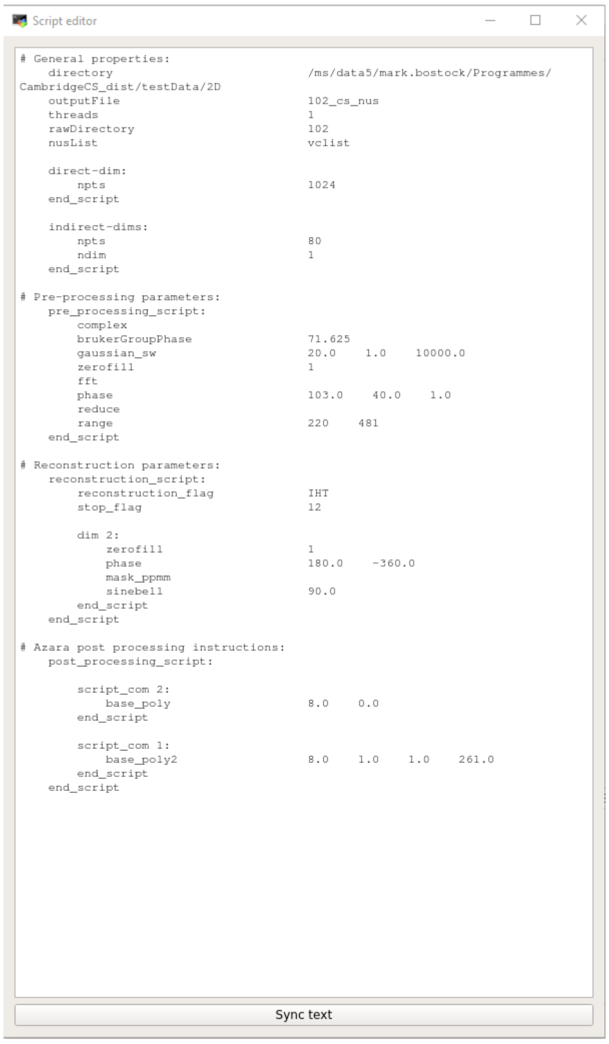
Running the reconstruction

At the stage you should save the script, using the buttom 'Save file'. Then click 'Process current script' to start the reconstruction. A second window will appear which gives some output describing the reconstruction and shows the progress of the reconstruction (Figure 4). A progress bar at the bottom of the main GUI window also shows the reconstruction progress (note that this only refers to the 'Reconstruction script' section and does not account for pre- or post-processing commands).

At the end of the reconstruction, the 'Graph plot' window displays the result (Figure 5).

Basic manipulations of the spectrum can be carried out here to check the result:

- Zoom: Shift + drag *or* Ctrl + drag
- Double-click: Reset display window
- Click and drag: Move spectrum (for a zoomed region only).
- $\uparrow\downarrow$: Change the base level up or down by the scale factor (LHS).
- 'd' 'f': Scale the 1D slice down or up.
- 'Number of levels': Change the number of displayed contour levels.
- 'Scale factor' (RHS): Change the scale factor between the contour levels.
- Single-click: freezes the position of the 1D slices, which can be useful for phasing. A second click causes the slices to follow the mouse position again.



```
# General properties:
    directory                          /ms/data5/mark.bostock/Programmes/
CambridgeCS_dist/testData/2D
    outputFile                         102_cs_nus
    threads                            1
    rawDirectory                       102
    nusList                            vclist

    direct-dim:
        npts                           1024
    end_script

    indirect-dims:
        npts                           80
        ndim                           1
    end_script

# Pre-processing parameters:
    pre_processing_script:
        complex
        brukerGroupPhase               71.625
        gaussian_sw                    20.0    1.0    10000.0
        zerofill                       1
        fft
        phase                          103.0    40.0    1.0
        reduce
        range                          220    481
    end_script

# Reconstruction parameters:
    reconstruction_script:
        reconstruction_flag            IHT
        stop_flag                      12

        dim 2:
            zerofill                   1
            phase                      180.0    -360.0
            mask_ppmm
            sinebell                   90.0
        end_script
    end_script

# Azara post processing instructions:
    post_processing_script:

        script_com 2:
            base_poly                  8.0    0.0
        end_script

        script_com 1:
            base_poly2                 8.0    1.0    1.0    261.0
        end_script
    end_script
```

Figure 3: Cambridge CS script for processing the 2D HSQC, experiment number 102, described in section 4.1. The script is equivalent to the commands shown in the GUI, Figure 2

- 'Phasing': Opens another window which allows the phasing to be altered in the direct and indirect dimensions. Note any changes to the phasing must be added manually to the processing script.

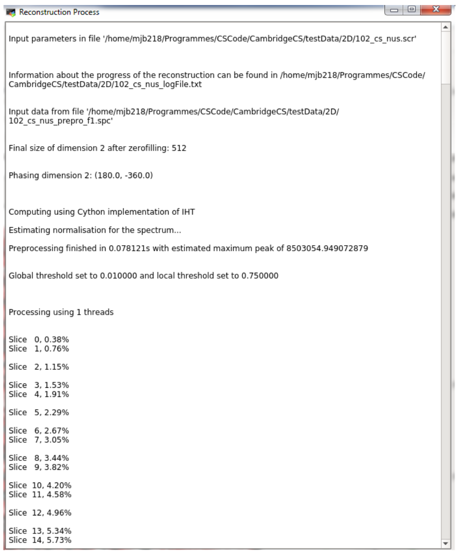
4.2 3D processing from scratch

4.2.1 3D [^{1}H,^{15}N]-HNCO

Change to the directory *~/testData/3D/HNCO*. This contains a 3D SOFAST(BEST)-[^{1}H,^{15}N]-TROSY HNCO experiment for the chicken α-spectrin SH3 domain, recorded with 23% sampling. This uses P/N type frequency discrimination in dimension 2 (gradient selection) and States-TPPI frequency discrimination in dimension 3.

Try setting up the processing, using the Cambridge CS GUI, or the scripting option noting the following:

- There are two indirect dimensions (^{15}N and ^{13}C) so set 'ndim' to 2. Under reconstruction parameters, this will then provide processing parameters for dim 2 and dim 3.

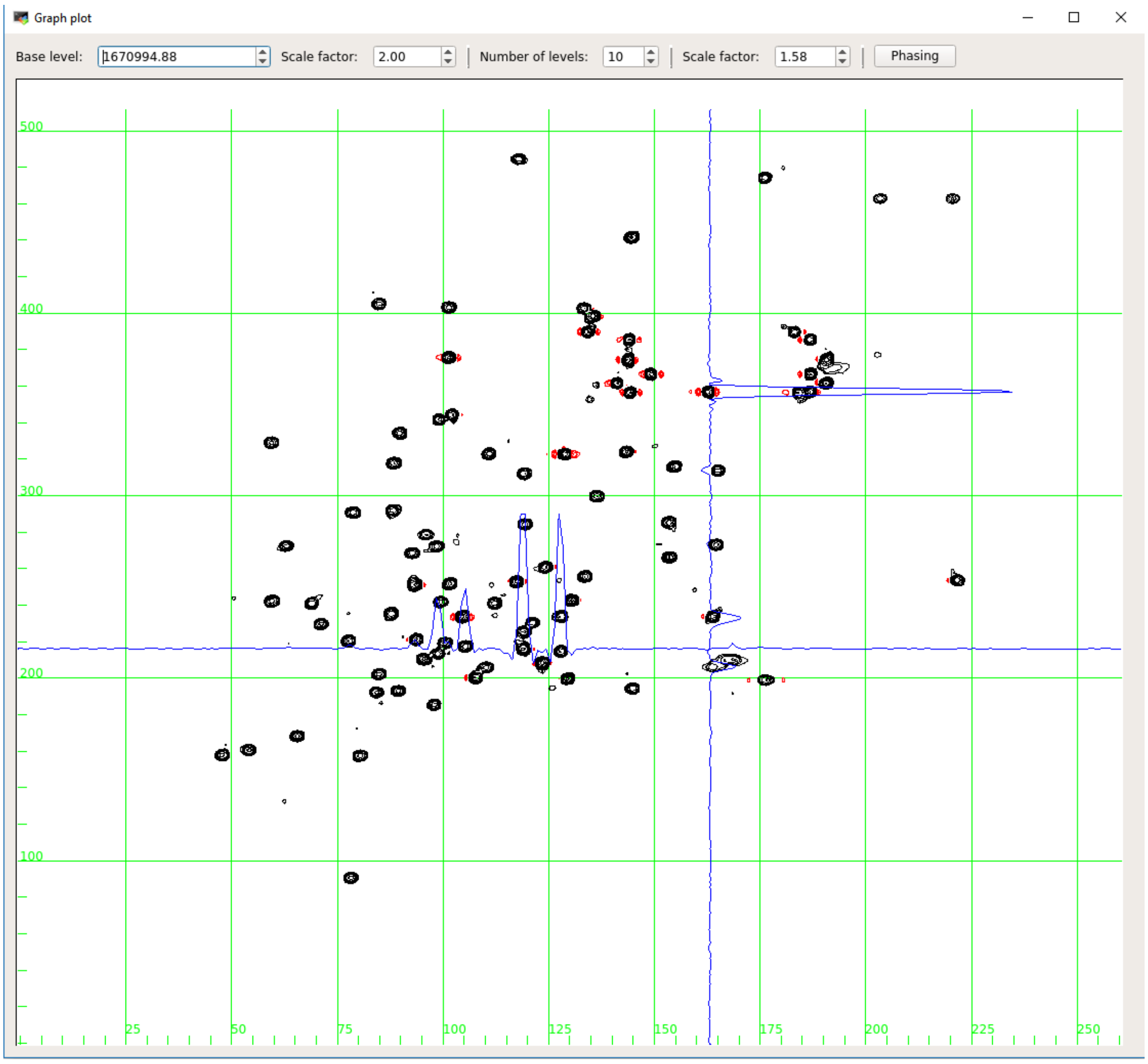


```
Input parameters in file '/home/mjb218/Programmes/CSCode/CambridgeCS/testData/2D/102_cs_nus.scr'


Information about the progress of the reconstruction can be found in /home/mjb218/Programmes/CSCode/
CambridgeCS/testData/2D/102_cs_nus_logFile.txt


Input data from file '/home/mjb218/Programmes/CSCode/CambridgeCS/testData/2D/
102_cs_nus_prepro_f1.spc'


Final size of dimension 2 after zerofilling: 512


Phasing dimension 2: (180.0, -360.0)



Computing using Cython implementation of IHT

Estimating normalisation for the spectrum...

Preprocessing finished in 0.078121s with estimated maximum peak of 8503054.949072879


Global threshold set to 0.010000 and local threshold set to 0.750000



Processing using 1 threads


Slice   0, 0.38%
Slice   1, 0.76%

Slice   2, 1.15%

Slice   3, 1.53%
Slice   4, 1.91%

Slice   5, 2.29%

Slice   6, 2.67%
Slice   7, 3.05%

Slice   8, 3.44%
Slice   9, 3.82%

Slice  10, 4.20%
Slice  11, 4.58%

Slice  12, 4.96%

Slice  13, 5.34%
Slice  14, 5.73%
```

Figure 4: Cambridge CS GUI reconstruction window showing progress of a CS reconstruction for experiment 102.

- Dim 2 (^{15}N) uses P/N-type (Echo/Antiecho) processing. To avoid a phase-twist lineshape, the echo and anti-echo modulated data needs to be converted to cosine/sine data. This is done using the *interlace 2* command, found under 'indirect-dims > Extra Params', where 2 indicates conversion of dimension 2.
- In the pre-processing script, *brukerGroupPhase* should be set to 71.625 for this dataset.
- Phasing for the indirect dimensions is (90,0) and (0,0) for dimensions 2 and 3 respectively. Use the *scaleT0* parameter ('Extra Params > scaleT0') to scale the first point by 0.5.
- In the post-processing script reverse dimensions 2 and 3 and apply baseline correction in all three dimensions.
- Check the processing first using FT reconstruction.
- CS reconstruction takes a little longer for a 3D (depending on the speed of your computer). You can force a faster (although less accurate reconstruction) by changing the global threshold. 'Reconstruction script > Extra Params > global threshold'. Try setting this to 1.
- Reconstruction speed can be substantially increased by changing the number of threads available to Cambridge CS. The programme uses Python's multiprocessing module. Select 'Data parameters > Threads'. Setting threads to 0 uses the maximum number of threads available on your system (note that this will subsantially slow down other software you may be running). Otherwise, if you know the number of threads available this can be set manually.

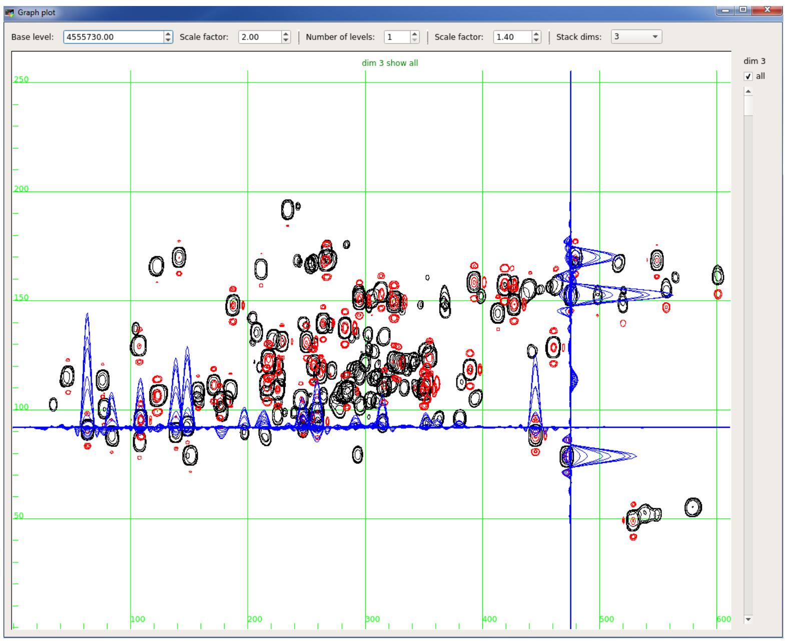
3D Graph Plot



Graph plot

Base level: 1670994.88 Scale factor: 2.00 | Number of levels: 10 | Scale factor: 1.58 | Phasing

Figure 5: Cambridge CS spectrum display window showing the result of IHT reconstruction of a 30% sampled 2D HSQC spectrum for ubiquitin.


Base level: 4555730.00 Scale factor: 2.00 | Number of levels: 1 | Scale factor: 1.40 | Stack dims: 3

dim 3 ✓ all

Figure 6: Cambridge CS spectrum display window showing the result of IHT reconstruction of a 23% sampled 3D SOFAST HNCO spectrum.

Once the processing has finished, the 'Graph plot' window is displayed. In addition to the options available previously, there is also a drop-down menu 'Stack dims'. The viewer displays two dimensions with the third dimension 'stacked'. The drop-down menu allows the stacked dimension to be changed from dim 3 (^{13}C) to dim 2 (^{15}N). Underneath the 'Stack dims' menu is a slider bar. When the tickbox 'all' is selected all the planes in the stacked dimension are displayed on top of each other. If this is unticked, the slider then allows one to scroll through the planes in the stacked dimension. The slice shown is indicated at the top of the spectrum view window.

- Try stacking dim 2 and dim 3. If dim 2 is stacked, the window displays dim 1 (^{1}H, x) vs dim 3 (^{13}C, y). If dim 3 is stacked the window displays dim 1 (^{1}H, x) vs dim 2 (^{15}N, y).
- Scroll through the different planes. It may be useful to change the number of levels (e.g. to 10) and lower the base level.

4.2.2 3D ^{13}C-NOESY

A 3D ^{13}C-NOESY spectrum of sensory rhodopsin II (pSRII), with 25% sampling, is available in the folder *~/testData/3D/13CNOESY*. Familiarise yourself with the script used for this spectrum, *1734_cs.scr*. Note that States-TPPI frequency discrimination is used in both indirect dimensions. Try processing this with FT reconstruction or one of the CS algorithms.

4.3 Pre- and post-processing using NMRPipe

Although Cambridge CS supports a full suite of pre- and post-processing functions based on the Azara processing package, many NMR users will wish to use the programme in combination with NMRPipe [19] for pre- and post-processing. As mentioned in section 3.5, NMRPipe takes the place of the pre- and post-processing modules. Cambridge CS still requires a processing script containing the 'Data parameters' and 'Reconstruction' sections.

The following sections show how to use Cambridge CS with NMRPipe.

4.3.1 2D Cambridge CS-Pipe processing

Change to the directory *~/testData/2D/102*. The NMRPipe script *fid.com* can be created using the *bruker* utility via the command *bruker -nus -sample nuslist* to convert the raw data from Bruker format to NMRPipe data format. This uses the module *nusExpand.tcl* to replace the missing data points with zeros, based on the sampled points listed in the nuslist. This module is discussed in more detail in the lecture/practical on NMRPipe. These 'missing' points can then be reconstructed later using the reconstruction algorithm. Using the 'nusExpand.tcl' module ensures that the NMRPipe headers are all set correctly. There is no need to create a mask using 'nusExpand.tcl'. The file *fid.com* should look as follows:

```
#!/bin/csh

nusExpand.tcl -mode bruker -sampleCount 40 -off 1 \
 -in ./ser -out ./ser_full -sample vclist

bruk2pipe -in ./ser_full \
  -bad 0.0 -ext -noaswap -AMX -decim 16 -dspfvs 12 -grpdly 0  \
  -xN              1024  -yN               256  \
  -xT               512  -yT               128  \
  -xMODE            DQD  -yMODE        Complex  \
  -xSW        10000.000  -ySW         2000.000  \
  -xOBS         500.012  -yOBS          50.672  \
  -xCAR           4.679  -yCAR             120  \
  -xLAB              HN  -yLAB             15N  \
  -ndim               2  -aq2D         Complex  \
  -out ./fid/test.fid -verb -ov

sleep 5
```

A few points to note:

- '-sampleCount' refers to the number of recorded samples in the vclist. This is the number of complex data points actually recorded in the indirect dimensions.
- '-sample' points to the nuslist for the relevant data set.
- 'nusExpand.tcl' writes out a new serial file with missing points replaced with zeros. Make sure to give this a new filename. Here we use 'ser_full'.
- In the 'bruk2pipe' conversion script make sure to read in ('-in') the zero-padded serial file ('ser_full'). Since this now has the equivalent size to a fully-sampled spectrum with the same t_{1max} as indicated in the nuslist, '-yN' is set to the full size.

Processing is then carried out using the script *nusproc.com*:

```
#!/bin/csh

set csout = 102_nus

nmrPipe -in ./fid/test.fid \
| nmrPipe  -fn SOL                                    \
| nmrPipe  -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5     \
| nmrPipe  -fn ZF -auto                                \
| nmrPipe  -fn FT                                      \
| nmrPipe  -fn PS -p0 237  -p1 33 -di                  \
| nmrPipe  -fn EXT -x1 220 -xn 480 -sw -verb           \
   -ov -out ./ft/test.nus

set dSize = `getParm -in ./ft/test.nus -parm NDSIZE -dim CUR_XDIM -fmt %.0f`

cambridgecs -a ./102_cs_iht_pipe.scr --directory ./cs --nuslist vclist \
               --outputFile {$csout}_pipe --threads 16 \
               --dataInput ./ft/test.nus --rawDirectory . --npts0 $dSize

set ftinput = {$csout}_pipe.ft1

nmrPipe -in ./cs/$ftinput -verb \
| nmrPipe -fn TP \
| nmrPipe -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5 \
| nmrPipe -fn ZF -zf 1 -auto                      \
| nmrPipe -fn FT                                  \
| nmrPipe -fn PS -p0 180 -p1 360 -di               \
| nmrPipe -fn TP \
    -ov -out ./ft2/test_cs.ft2
```

The direct and any indirect dimensions are processed using standard NMRPipe commands. The output from 'fid.com' is passed as the input to the NMRPipe stream. CS reconstruction of the indirect dimensions is carried out by the Cambridge CS code. The command for this comes immediately after the direct dimension processing. A few important points to note from the above script:

- This script reads in the NMRPipe format data, *test.fid*, and applies standard NMRPipe commands for 1D data processing. However, an important difference with standard NMRPipe processing is that Cambridge CS processes data in place, i.e. it is not necessary to transpose successive dimensions to the x-axis for processing. Hence the line:

  ```
  | nmrPipe  -fn TP \
  ```

  at the end of the direct dimension part of the NMRPipe script is ***not*** needed, and instead is placed after the CS processing. This is different to the usual NMRPipe convention.

- Cambridge CS needs to receive the correct number of direct dimension points - this may of course be changed by the NMRPipe command 'nmrPipe -fn EXT'. Therefore the parameter 'dSize' is set to the number of direct dimension points after processing, using the line 'getParm -in test.nus -parm NDSIZE -dim CUR_XDIM -fmt %.0f'

- Cambridge CS is then called via the command ‘cambridgecs -a’ followed by the name of the reconstruction script ‘102_cs_iht_pipe.scr’ (see below). A few command line options allow variables in the ‘.scr’ file to be varied. This could, for example, allow batch processing.

```
 --directory    output directory for processed data
--nuslist       NUS list
--outputFile    stem for output filenames
--threads       number of threads (CPU cores) to use
--dataInput     Pipe file from output of direct dimension processing
--rawDirectory  Directory containing raw data
--npts0         Number of points in the direct dimension,
                  set from the parameter ‘dSize’.
```

- The result of CS reconstruction of the indirect dimensions is time-domain data for the indirect dimensions, with the missing data points reconstructed using one of the CS algorithms included with the Cambridge CS package. This is then passed to the remainder of the Pipe script for further FT processing, as for a normal dataset.

The script file used by the command ‘cambridgecs -a ./102_cs_iht_pipe.scr ...’ is very similar to the scripts we set up earlier in the practical. You can view the script by reading the script file *102_cs_iht_pipe.scr* into Cambridge CS using the command:

```
cambridgecs -g 102_cs_iht_pipe.scr
```

The setup is similar to scripts we have looked at earlier, however, there is no ‘Pre-processing script’ as this has already been carried out using NMRPipe. There are also a few changes in the ‘Data Parameters’ section. In addition to the usual commands we also specify:

- Data Input: this points to the output from NMRPipe processing in the direct dimension, *test.nus*. Note this is also set by the flag ‘–dataInput’ which will overwrite any value in the script.

- Data input flag: select nmrPipe.

- Header: NMRPipe data files have a header of length 512 bits. 512 is set as the default value so there should be no need to alter this.

- Open the ‘Script editor’ to see how these commands are included in the processing script.

The NMRPipe format data is read into the Cambridge CS data format with the above commands and can then be used in the ‘Reconstruction script’.

- Process the data using the NMRPipe script and view the data using NMRDraw. You can Also comment out the CS processing line, which will give pure ‘FT’ processing, i.e. no data reconstruction and compare the FT vs. CS reconstructions.

4.3.2 3D Cambridge CS-Pipe processing

HNCO

The NMRPipe processing workflow can be used for higher dimensional spectra. Additional examples are available for the 3D spectra. Change to ∼ */testData/3D/HNCO/550*, which contains NMRPipe scripts for the HNCO processed earlier. Open *fid.com* which reads the Bruker data into the NMRPipe format:

```
#!/bin/csh

#This line may be necessary if the NMRPipe macros are not
#found on the system path.
#Replace with the relevant path for your system.
set NMRTXT=/programs/x86_64-linux/nmrpipe/20181211/nmrtxt

nusExpand.tcl -mode bruker -sampleCount 350 -off 1 \
 -in ./ser -out ./ser_full -sample vclist_coord

bruk2pipe -in ../550/ser_full \
  -bad 0.0 -noaswap -DMX -decim 16 -dspfvs 12 -grpdly -1  \
  -xN              1024  -yN                64  -zN               96  \
  -xT               512  -yT                32  -zT               48  \
  -xMODE            DQD  -yMODE        Complex  -zMODE       Complex  \
  -xSW        10000.000  -ySW         2530.364  -zSW        1666.667  \
  -xOBS         500.132  -yOBS          50.678  -zOBS        125.757  \
  -xCAR             4.7  -yCAR         120.000  -zCAR        176.000  \
  -xLAB              HN  -yLAB             15N  -zLAB            13C  \
  -ndim               3  -aq2D         Complex \
| nmrPipe -fn MAC -macro $NMRTXT/ranceY.M -noRd -noWr   \
| pipe2xyz -x -out ./fid/test%03d.fid -verb -ov

sleep 5
```

In this example, 350 (^{15}N, ^{13}C) complex pairs are recorded in the indirect dimensions, hence '-sampleCount' is set to 350. The full spectrum would contain 64×96 real + imaginary points in the indirect dimensions, so after expansion by 'nusExpand.tcl', the 'bruk2pipe' script is setup with the full data size. In this particular example, the ^{15}N dimension (dimension 2) is acquired using Echo/Anti-echo selection and must be converted to cosine/sine type data for processing. For this, the NMRPipe macro *ranceY.M* is used, which applies Cavanagh-Rance-Kay mode conversion in dimension 2 using the line:

```
| nmrPipe -fn MAC -macro $NMRTXT/ranceY.M -noRd -noWr   \
```

The equivalent macro *ranceZ.M* would apply Cavanagh-Rance-Kay mode conversion in dimension 3 although that is not required in this case.
The data processing script, nusproc.com is shown below:

```
#!/bin/csh
```

```
set csout = 550_nus

xyz2pipe -in fid/test%03d.fid -x  \
| nmrPipe  -fn SOL                                  \
| nmrPipe  -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5  \
| nmrPipe  -fn ZF -auto                             \
| nmrPipe  -fn FT                                   \
| nmrPipe  -fn PS -p0 241  -p1 0.0 -di               \
| nmrPipe  -fn EXT -left -sw                        \
 -ov -out ./ft/test.nus

set dSize = `getParm -in ./ft/test.nus -parm NDSIZE -dim CUR_XDIM -fmt %.0f`

cambridgecs -a ./550_pipe_cs.scr --directory ./cs --nuslist vclist_coord \
                 --outputFile {$csout}_pipe --threads 16 \
                 --dataInput ./ft/test.nus --rawDirectory . --npts0 $dSize

set ftinput = {$csout}_pipe.ft1

xyz2pipe -in ./cs/$ftinput -x \
| nmrPipe -fn TP \
| nmrPipe -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5 \
| nmrPipe -fn ZF -zf 1 -auto                      \
| nmrPipe -fn FT                                  \
| nmrPipe -fn PS -p0 -90 -p1 0 -di               \
| nmrPipe -fn TP                            \
| nmrPipe -fn ZTP \
| nmrPipe -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5 \
| nmrPipe -fn ZF -zf 1 -auto                      \
| nmrPipe -fn FT              \
| nmrPipe -fn PS -p0 0 -p1 0 -di                      \
| nmrPipe -fn ZTP             \
    -ov -out ./ft3/test_cs.ft3
```

- Run *fid.com*. The output is written as a series of 2D planes to the directory *∼/fid*.
- Open the file *nusproc.com*. This starts by processing the direct dimension for each plane in *∼/fid*. The output is written to the file *test.nus*. Note that Cambridge CS processing requires a full multidimensional data set for processing rather than individual planes.
- Note also that as described above for the 2D dataset, no transpose command (`nmrPipe -fn TP`) should be included before the CS reconstruction of the indirect dimensions. This command is applied after CS processing.
- Read the script file 550_pipe_cs.scr into Cambridge CS. Note that unlike *fid.com*, Cambridge CS data treats multidimensional data as a pseudo-2D. Consequently the total number of points in the indirect dimensions is specified ($6144 = 64 \times 96$).
- Try processing the data using the NMRpipe script. Note that the output is a single 3D spectrum file. You can view this in CCPN Analysis. Alternatively, you could

write the output as a series of 2D planes, with the command `| pipe2xyz -x -out ./ft3/test%03d.ft3 -verb -ov`

3D ^{13}C NOESY

Example NMRPipe scripts are also available for the ^{13}C NOESY in the directory $\sim$ */testData/3D/13CNOESY/1734*. The setup is comparable to the 3D HNCO described above except that there is no requirement for the Rance-Kay conversion since the indirect dimensions both use States-TPPI quadrature detection. These scripts are shown below: NMRPipe conversion script, `fid.com`

```
#!/bin/csh

nusExpand.tcl -mode bruker -sampleCount 480 -off 1 \
 -in ser -out ser_full -sample ../1734/vclist_coord

bruk2pipe -in ser_full \
  -bad 0.0 -aswap -DMX -decim 2000 -dspfvs 20 -grpdly 67.9862518310547  \
  -xN              1024  -yN                60  -zN               128  \
  -xT               512  -yT                30  -zT                64  \
  -xMODE            DQD  -yMODE        Complex  -zMODE        Complex  \
  -xSW        10000.000  -ySW         3263.708  -zSW             4000  \
  -xOBS          800.13  -yOBS         800.13   -zOBS         201.197  \
  -xCAR            4.72  -yCAR            0.75  -zCAR          21.989  \
  -xLAB             1Hx  -yLAB              1H  -zLAB             13C  \
  -ndim               3  -aq2D          States                         \
| pipe2xyz -x -out ./fid/test%03d.fid -verb -ov -to 0
```

NMRPipe processing script, `nusproc.com`

```
#!/bin/csh

set csout = 1734_nus

xyz2pipe -in fid/test%03d.fid -x \
| nmrPipe -fn SOL \
| nmrPipe  -fn SP -off 0.5 -end 0.98 -pow 2  \
| nmrPipe  -fn ZF -auto                          \
| nmrPipe  -fn FT -verb                             \
| nmrPipe  -fn PS -p0 282  -p1 0 -di              \
| nmrPipe  -fn EXT -left -sw          \
 -ov -out ./ft/test.nus

set dSize = `getParm -in ./ft/test.nus -parm NDSIZE -dim CUR_XDIM -fmt %.0f`

cambridgecs -a ./1734_pipe_cs.scr --directory ./cs --nuslist vclist_coord \
              --outputFile {$csout}_pipe --threads 16 \
              --dataInput ./ft/test.nus --rawDirectory . --npts0 $dSize
```

```
set ftinput = {$csout}_pipe.ft1

xyz2pipe -in ./cs/$ftinput -x \
| nmrPipe -fn TP \
| nmrPipe -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5 \
| nmrPipe -fn ZF -zf 1 -auto                       \
| nmrPipe -fn FT                                   \
| nmrPipe -fn PS -p0 -45 -p1 0 -di                \
| nmrPipe -fn TP                             \
| nmrPipe -fn ZTP \
| nmrPipe -fn SP -off 0.5 -end 0.98 -pow 2 -c 0.5 \
| nmrPipe -fn ZF -zf 1 -auto                       \
| nmrPipe -fn FT             \
| nmrPipe -fn PS -p0 0 -p1 0 -di                      \
| nmrPipe -fn ZTP             \
    -ov -out ./ft3/test_cs.ft3
```

- Try running the 13C NOESY scripts. You may also want to compare with the complete Cambridge CS based processing.

4.4 Other processing options

4.4.1 Undersampling a fully-sampled spectrum

This can be used to test different sampling schedules on an existing fully sampled spectrum. Set up the reconstruction as described in section 4.1, but set 'Raw data directory' to *102_full*, which contains fully-sampled raw data. In 'Data parameters > Extra parameters', choose 'Original NUS list' and select *∼/102_full/vclist_full.* This list describes the points in the fully-sampled spectrum. Choose the undersampled NUS list, *vclist* for the 'NUS list' parameter. The reconstruction should be run as before. The full script is contained in the file *102_ft_full.scr*. The 'Data parameters' section of the script should look as follows:

```
# General properties:
    directory                           ~/CambridgeCS/testData/2D
    outputFile                          102_ft_full
    threads                             1
    rawDirectory                        102_full
    nusList                             vclist
    nusListOriginal                     102_full/vclist_full

    direct-dim:
        npts                            1024
    end_script

    indirect-dims:
        npts                            256
        ndim                            1
    end_script
```

- Test the effects of different undersampling schedules by modifying the vclist: Copy *~102_full/vclist* and delete some lines from the list. Rerun the reconstruction and compare the difference. (NB The 'Graph plot' window can be called directly from the command line with `graphPlot filename_block.spc.par`).

4.4.2 Processing a fully-sampled spectrum with no vclist

Cambridge CS can also be used to process fully-sampled data without the need for an NUS list. Setup the script as in section 4.1, using '102_full' for the 'Raw data directory'. In the 'Data parameters' section:

- Remove the 'NUS list' parameter: either delete the line from the script editor window and press 'Sync text' or untick 'NUS list' under 'Data parameters > Extra Params'.
- Select 'Extra Params > Uniform sampling flag'.
- In 'Reconstruction script' select 'dim 2 > Extra Params > max_points' and input 256 as the maximum number of points.
- Look in the 'Script Editor' to see the commands that have been inserted. Once you are familiar with the scripting language these changes can easily be made directly in the script rather than using the GUI.
- Change the reconstruction type to 'FT' and run the reconstruction as usual. (A fully-sampled data set can also be processed with a CS algorithm and will give the same result as FT processing).

4.4.3 Linear prediction

CS reconstructions are typically used to 'fill-in' missing data points in an NUS experiment. However, the algorithms are also suitable for extending datasets in a manner similar to linear prediction. In fact this was one of the earliest identified uses of the iterative soft thresholding algorithm (IST) [23].

Change to the directory *~/testData/LP/*. Here the HNCO experiment used earlier is processed with the Cambridge CS LP function. This particular experiment uses a constant-time ^{15}N dimension (dimension 2), which needs a strongly decaying window function in order to prevent truncation artifacts. The experiment has 32 points in the ^{15}N dimension, so we decide to double the length of this dimension.

- Under 'Reconstruction script > dim 2' choose 'Extra Params' and 'extend'. Set 'extend' to 32. Extend takes a value for the number of ***complex*** points by which to extend the data. Although this is different to the general convention in Cambridge CS of expressing total number of real and imaginary points, this avoids the risk of the user specifying an odd number of points, which would of course be incorrect if counting in total number of real and imaginary points.
- Try switching the *extend* option on and off. Note that you can keep existing 'Graph plot' windows open. You will see the effect on resolution by extending the data. Note also that you will need *Stack dims* set to 3 in order to display the ^{1}H,^{15}N plane.
- To explore further, try switching off the weighting function in dimension 2. Run with and without the extend function. Without CS-extend you will observe truncation artifacts from the non-decaying constant time data.

4.4.4 Incomplete experiments

Sometimes an experiment may be stopped early for various reasons. In Bruker data, the *ser* file will still have the size of the full experiment, however, points after the experiment was stopped will not be correct. In this case when processing the data, it is useful to specify which points were correctly recorded. The *nuslist* is still needed in order to rearrange the data into the correct format for processing within the Cambridge CS software, however, we do not want to use the unrecorded points in the reconstruction itself as this will distort the result.

Change to the directory *∼/testData/Incomplete/*. Spectrum *667* is a SOFAST ^{1}H^{13}C HMQC which should have been recorded with 120 points (total) in the ^{13}C dimension; however the experiment crashed after only 87 points.

- Look at the script *667_cs.scr*. An additional argument, *acquired_points* is included which is set to 87 here i.e. the total number of recorded points. Other parameters are set to the normal values.
- Try running this script with and without the acquired points flag set. You will quickly see the difference. Without the *acquired_points* flag, the reconstruction uses the full data set including the last 33 points that were wrong.
- This example demonstrates the utility of randomising the *nuslist* order. In this case a linear order was used for the *nuslist* so some resolution will be lost as the longest time-points were not recorded.

5 Acknowledgements

Many thanks to Rob Tovey who worked extensively on the code in its current form, Wayne Boucher for the Pre-processing script and useful discussions about NMR data, Daniel Holland, Daniel Nietlispach, and various people in Cambridge who have tested the code so far.

6 Software

The software and examples are available if you are interested.
Please e-mail `mark.bostock@helmholtz-muenchen.de` or `dn206@cam.ac.uk` if you would like a copy.

Appendices

A Lookup table for Bruker Avance data

Lookup values are given in the main body of the table for a particular (DSPFVS, DECIM) coordinate. Using the lookup value from the table below, GRPDLY = 0.5 × *lookup*/DECIM. The value of GRPDLY is provided as an input to **brukerGroupPhase**. This is equivalent to a first order phase correction of 360×<GRPDLY>.

DSPFVS	10	11	12	13	20
DECIM					
2	179	184	184	11	-1
3	201	219	219	17	-1
4	533	384	384	23	-1
6	709	602	602	35	-1
8	1097	852	852	47	-1
12	1449	1668	1668	71	-1
16	2225	2312	2292	95	-1
24	2929	3368	3368	143	-1
32	4481	4656	4616	191	-1
48	5889	6768	6768	287	-1
64	8993	9344	9264	383	-1
96	11809	13568	13568	575	-1
128	18017	18560	18560	-1	-1
192	23649	27392	27392	-1	-1
256	36065	36992	36992	-1	-1
384	47329	55040	55040	-1	-1
512	72161	73856	73856	-1	-1
768	94689	110336	110336	-1	-1
1024	144353	147584	147584	-1	-1
1536	189409	220928	220928	-1	-1
2048	288737	295040	295040	-1	-1
2080	-1	-1	-1	-1	282814

References

[1] J. C. J. Barna, E. D. Laue, M. R. Mayger, J. Skilling and S. J. P. Worrall, *J. Magn. Reson.*, 1987, **73**, 69–77.

[2] B. E. Coggins and P. Zhou, *J. Biomol. NMR*, 2008, **42**, 225–239.

[3] K. Kazimierczuk, W. Koźmiński and I. Zhukov, *J. Magn. Reson.*, 2006, **179**, 323–328.

[4] D. Marion, *J. Biomol. NMR*, 2005, **32**, 141–150.

[5] V. Y. Orekhov, I. V. Ibraghimov and M. Billeter, *J. Biomol. NMR*, 2001, **20**, 49–60.

[6] D. Rovnyak, D. P. Frueh, M. Sastry, Z.–Y. J. Sun, A. S. Stern, J. C. Hoch and G. Wagner, *J. Magn. Reson.*, 2004, **170**, 15–21.

[7] P. Schmieder, A. Stern, G. Wagner and J. Hoch, *J. Biomol. NMR*, 1993, **3**, 569–576.

[8] P. Schmieder, A. S. Stern, G. Wagner and J. C. Hoch, *J. Biomol. NMR*, 1994, **4**, 483–490.

[9] V. Tugarinov, L. E. Kay, I. V. Ibraghimov and V. Y. Orekhov, *J. Am. Chem. Soc.*, 2005, **127**, 2767–2775.

[10] M. R. Palmer, C. L. Suiter, G. E. Henry, J. Rovnyak, J. C. Hoch, T. Polenova and D. Rovnyak, *J. Phys. Chem. B*, 2015, **119**, 6502–6515.

[11] B. F. Logan, Columbia University, New York, 1965.

[12] D. L. Donoho, *IEEE T. Inform. Theory.*, 2006, **52**, 1289–1306.

[13] E. J. Candès, J. Romberg and T. Tao, *IEEE T. Inform. Theory.*, 2006, **52**, 489–509.

[14] M. Lustig, D. L. Donoho and J. M. Pauly, *Magn. Reson. Med.*, 2007, **58**, 1182–1195.

[15] D. J. Holland, M. J. Bostock, L. F. Gladden and D. Nietlispach, *Angew. Chem. Int. Ed.*, 2011, **50**, 6548–6551.

[16] K. Kazimierczuk and V. Y. Orekhov, *Angew. Chem. Int. Ed.*, 2011, **50**, 5556–5559.

[17] S. G. Hyberts, A. G. Milbradt, A. B. Wagner, H. Arthanari and G. Wagner, *J. Biomol. NMR*, 2012, **52**, 315–327.

[18] M. J. Bostock, D. J. Holland and D. Nietlispach, *J. Biomol. NMR*, 2012, **54**, 15–32.

[19] F. Delaglio, S. Grzesiek, G. W. Vuister, G. Zhu, J. Pfeifer and A. Bax, *J. Biomol. NMR*, 1995, **6**, 277–293.

[20] M. W. Maciejewski, M. Fenwick, A. D. Schuyler, A. S. Stern, V. Gorbatyuk and J. C. Hoch, *Proc. Natl. Acad. Sci. USA*, 2011, **108**, 16640–16644.

[21] A. D. Schuyler, M. W. Maciejewski, A. S. Stern and J. C. Hoch, *J. Magn. Reson.*, 2015, **254**, 121–130.

[22] M. J. Bostock, D. J. Holland and D. Nietlispach, *J. Biomol. NMR*, 2016, doi:10.1007/s10858-016-0062-9.

[23] A. S. Stern, D. L. Donoho and J. C. Hoch, *J. Magn. Reson.*, 2007, **188**, 295–300.