Electronic Supplementary Material (ESI) for Organic & Biomolecular Chemistry.

Resolving the complexity in human milk oligosaccharides using pure shift methods and CASPER

Electronic Supporting Information

Marshall J. Smith,[a] *Emma L. Gates,*[a] *Göran Widmalm,*[b] *Ralph W. Adams,*[a] *Gareth A. Morris*[a] *and Mathias Nilsson**[a]

[a] Department of Chemistry, The University of Manchester, Oxford Road, Manchester, M13 9PL, UK.

[b] Department of Organic Chemistry, Arrhenius Laboratory, Stockholm University, S-106 91 Stockholm, Sweden

* Correspondence to mathias.nilsson@manchester.ac.uk

Table of Contents

A. Experimental section

1. Details of pulse sequences

Schematic representations of the pulse sequences used in this study are shown in Figure S1: (a) F1-PSYCHE-TOCSY, (b) 1D selective pure shift TOCSY and (c) real-time BIRD multiplicity edited HSQC (edPSHSQC). Narrow black and grey rectangles represent hard 90° and 180° pulses, respectively. Shaped black pulses represent frequency-selective 180° pulses, typically RSNOB or REBURP. The white trapezoids with a single arrow represent low power 180° chirp pulses for zero quantum coherence suppression (ZQS). These pulses had durations of 10 ms and 30 ms and a 20 kHz bandwidth. White trapezoids with two arrows represent low flip angle (β) frequency-swept saltire pulses, typically with 20 kHz bandwidth, 30 – 120 ms duration and 20° flip angle. Grey trapezoids represent gradient pulses for enforcing the coherence transfer pathway (CTP). Wide grey rectangles represent weak field gradient pulses applied simultaneously with the ZQS chirp pulses or the saltire pulse to suppress zero-quantum coherences. All gradient pulses were followed by a stabilisation delay, typically with a duration of 1 ms. The DISPI-2 isotropic mixing duration required depends on the system under investigation; in this work durations of 120 – 200 ms were typically used.

The discontinuous FID in (b) represents the interferogram acquisition mode used in the selective PSYCHE-TOCSY experiment, giving a pseudo-2D homonuclear decoupled dataset. In interferogram acquisition, a short chunk of data of duration ($1/SW_1$), to minimise J_{HH} evolution during the chunks, is acquired for each t_1 increment. Typically, the duration of the data chunk was 20 ms (SW_1 = 50 Hz). J_{HH} was refocused in the middle of the chunk by setting τ_1 in (b) to $1/4SW_1$. The number of chunks acquired depends on the frequency resolution required, here 32 chunks were sufficient.

The edPSHQC experiments used real-time acquisition, with the acquisition of short chunks of data being periodically interrupted by the BIRD *J*-refocusing element. In this work, 16 chunks (*n*) with a 25.6 ms chunk duration (τ_{CH}) were acquired. The effective heteronuclear coupling constant, J_{CH}, was chosen to be 145 Hz. In pulse sequence (c) the τ_1 and τ_2 delays were set to $1/4^1J_{CH}$ and $1/2^1J_{CH}$, respectively. The ratio between the amplitudes of the gradient pulses marked by black trapezoids was set to the inverse of the ^{1}H and ^{13}C gyromagnetic ratios, 1:0.2515; 512 increments were acquired in the indirect dimension. To obtain high-resolution edHSQC and edPSHSQC spectra, spectral aliasing was used in the indirect dimension.

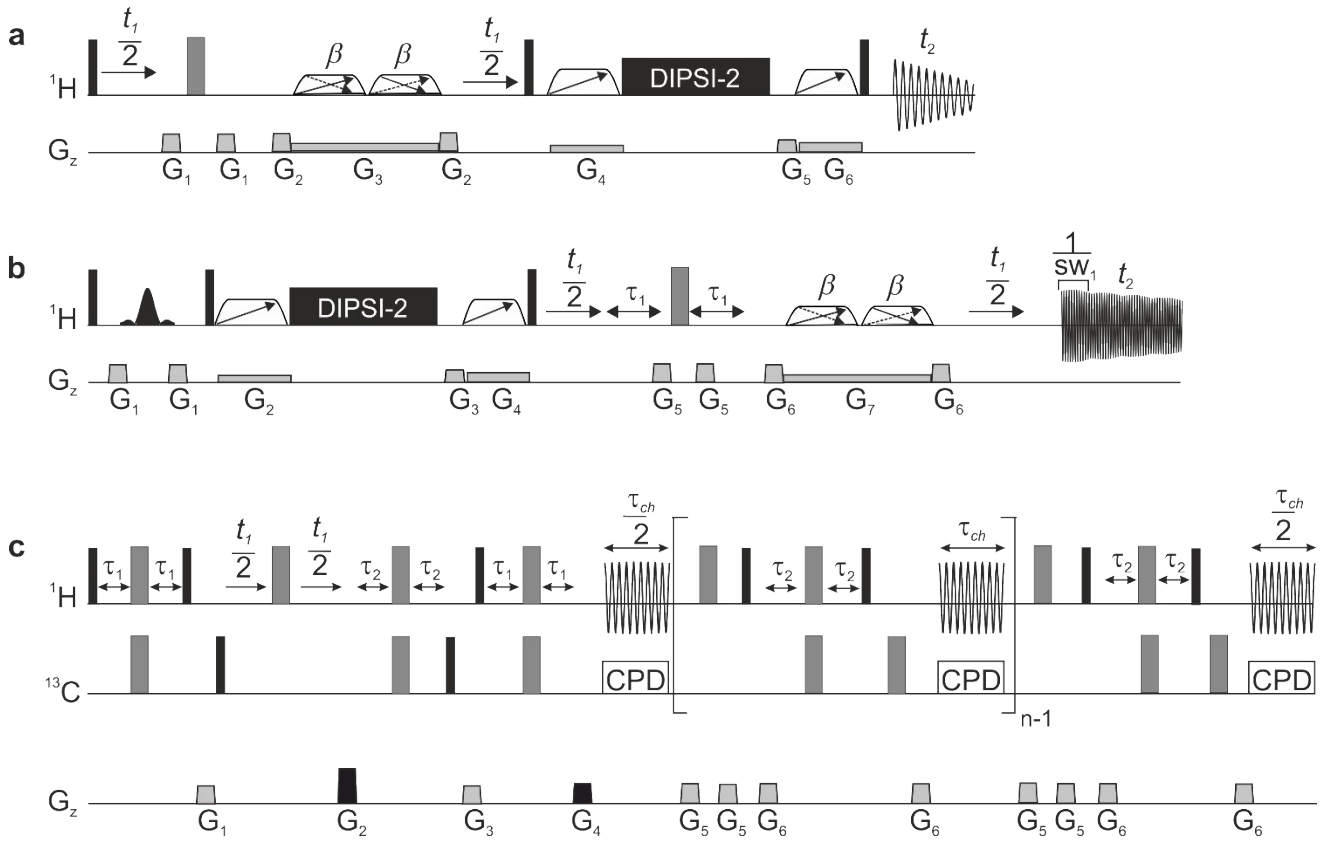

Figure S1. Schematic representation of the pulse sequences used in this work (a) F_1-PSYCHE-TOCSY, (b) pure shift selective TOCSY using a PSYCHE *J* refocusing element and (c) real-time multiplicity-edited pure shift HSQC (edPSHQC) using a BIRD *J* refocusing element. In all pulse sequences, the black and grey narrow rectangles represent hard 90° and 180° pulses, respectively. The shaped black pulse represents a frequency selective 180° pulse. In (a) and (b) the trapezoids with a single arrow represent chirp pulses and the trapezoids with a double arrow represent low flip angle saltire pulses. The grey trapezoids represent field gradient pulses to support the CTP selection. The wide grey rectangles are gradients applied simultaneously with the chirp pulses to aid in zero quantum coherence suppression.

2. Instructions for covariance processing F_1-PSYCHE-TOCSY data in Topspin

Following the initial standard processing (Fourier transformation, baseline correction, phase correction and apodization the steps below can be performed to obtain spectra that are pure shift in both dimensions.

1) Type “covariance man” into the command line.
2) Select the “within one file” option.
3) Use the square root of the covariance matrix.
4) Calculate covariance along F_2.
5) Use existing data.
6) Use the “no noise threshold” option.

3. Suggested scheme of work for using pure shift experiments in combination with CASPER

After acquiring the respective datasets, the pure shift HSQC and TOCSY peak-picking macros (section C) can be used to generate CASPER-compatible input data. The macros use Bruker standard parameters to control the region that is peak picked, the minimum and maximum intensity of the contours, and the signs of the signals to be identified.

In this work, both positive and negative contours were identified in the edPSHSQC spectrum. The methylene protons remain as doublets in the edPSHSQC spectrum and have opposite phases to methine and methyl protons. For any negative resonances identified, the peak-picking macro will determine whether any other signals have the same ^{13}C chemical shift within a user-defined limit. For two negative signals that have the same ^{13}C chemical shift, the macro will calculate the frequency difference between the two peaks and compare it to the maximum scalar coupling threshold (J_{max}) set by the user. If the frequency difference is less than J_{max} then a single chemical shift from the average frequency of the two signals is outputted. Hence, the residual doublets in the spectrum give a single ^{1}H and ^{13}C chemical shift for each chemical environment in an oligosaccharide. Finally, the edPSHSQC peak-picking macro has an option for calculating the ^{13}C chemical shifts for any peaks that were aliased into the spectrum.

CASPER also expects the ^{1}H chemical shifts in the edPSHSQC spectrum to be the same in pure shift TOCSY spectra. The F_1-PSYCHE-TOCSY peak-picking macro checks the signals identified in the PSTOCSY data against those in the edPSHSQC data. This requires that the two datasets are referenced consistently. If no peaks in the TOCSY data match those in the edPSHSQC, the macro will abort and inform the user that there may be a problem with referencing. Any ^{1}H signals in the PSTOCSY spectrum that do not have a corresponding ^{1}H signal in the edPSHSQC are removed from the outputted peak list. This minimises spurious peaks such as strong coupling artefacts in F_1 being misidentified, which would complicate assignment. In addition, for samples containing water, any correlations with the water resonance are automatically removed; this would otherwise require manually editing the peak list. CASPER only considers correlations from anomeric protons in TOCSY data, so the macro has been designed to check that ^{1}H signals in the indirect dimension (F_1) of PSTOCSY data have corresponding carbon chemical shifts in the edPSHSQC dataset that are within a user-defined chemical shift range. ^{1}H signals that do not have a corresponding ^{13}C chemical shift within this range are removed from the peak list, ensuring that only correlations from anomeric protons are identified. In addition, diagonal peaks in the F_1-PSYCHE-TOCSY spectrum are removed from the outputted peak list, as these are not used by CASPER.

The 1D pure shift selective TOCSY macro has the same functionalities as the F_1-PSYCHE-TOCSY peak-picking macro; in addition, the outputted peak list has a 2D format to be compatible with CASPER. As for the 2D PSTOCSY macro, the anomeric signal is identified by comparison to the PSHSQC data set, ensuring that the ^{1}H anomeric signal has a corresponding ^{13}C chemical shift within a user-defined chemical shift range. The outputted peak list then uses this chemical shift to generate a 2D style output in which the identified anomeric signal is correlated with the signals identified in the 1D spectrum. The diagonal peak analogue is removed from the output file.

CASPER is freely available for use at http://www.casper.organ.su.se/casper/. In this work, the "Determine Glycan Structure" module was used throughout. The 2D dataset option was selected in all cases. CASPER can either work with

unknown residues or, if the general structure of the molecule is known, then the residues and linkage types can be entered. The calculation time for structure determination is significantly longer if unknown residues are used. CASPER is limited to a 3-minute calculation time for chemical shift prediction and a 10-minute calculation time for structural elucidation, and time-out may occur if all unknown residues are used. In cases where mixtures of two anomeric forms are inputted into CASPER, the "Free anomer in reducing end" option can be selected. In this work, the α and β anomeric form data were inputted to CASPER separately. The pure shift HSQC and pure shift TOCSY data peak lists were inputted to CASPER from the files generated using the macros in section C. In all cases presented this was sufficient to discriminate between the top structure assignments produced by CASPER.

B. Experimental data

1. Lactose experimental data

108 mg of lactose (Kemikaliebolaget KEBO AB, Stockholm, Sweden) was dissolved in 700 μL D_2O, with TSP added as a reference. Figure S2 shows the gain in resolution that is achievable using pure shift methods; (a) the conventional ^1H spectrum and (b) the PSYCHE spectrum of this sample. The insets within the figure show expansions of the region 3.5 – 4.0 ppm; despite the significant increase in resolution achieved by pure shift experiments, some spectral overlap remains.

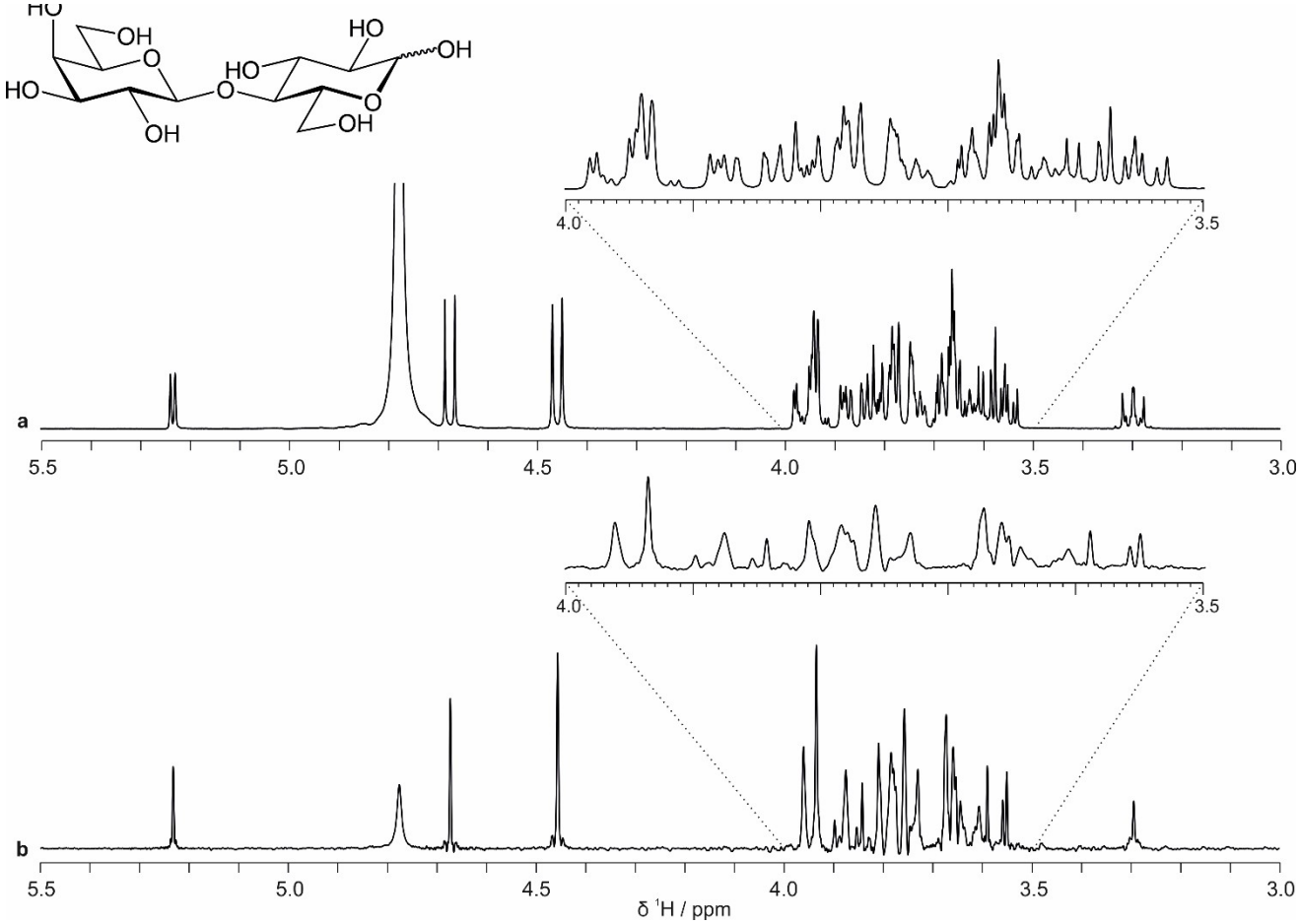

Figure S2. Comparison of (a) the conventional ^1H NMR spectrum and (b) the pure shift spectrum using a PSYCHE J_{HH} refocusing element, for lactose dissolved in D_2O. The insets show expansions of the region 3.5 – 4.0 ppm.

Figure S3 shows a comparison between the conventional HSQC (black and blue contours) and real-time pure shift HSQC (red and green contours) spectra, both with multiplicity editing, for a sample of 450 mM lactose in D_2O with TSP added as a reference. The edPSHSQC experiment used real-time acquisition to record homonuclear decoupled data in similar experiment times to the conventional edHSQC data. The BIRD homonuclear decoupling method was used, as the sensitivity penalty for selecting active spins (those bound directly to ^{13}C) has already been paid in the HSQC element of the sequence. Both edHSQC and edPSHSQC reduce some of the spectral overlap shown in Figure S2 by spreading the signals over an additional dimension. However, some spectral overlap remains in the conventional edHSQC spectrum, as shown by the overlapping black contours in the inset of Figure S3. The red contours, which represent the

edPSHSQC spectrum, show that by suppressing the homonuclear scalar coupling much higher resolution spectra can be recorded, minimising spectral overlap. The inset clearly shows that the multiplet structure is suppressed in the edPSHSQC spectrum, which allows automatic peak-picking routines to be used to generate CASPER-compatible data without the need for excessive line broadening.

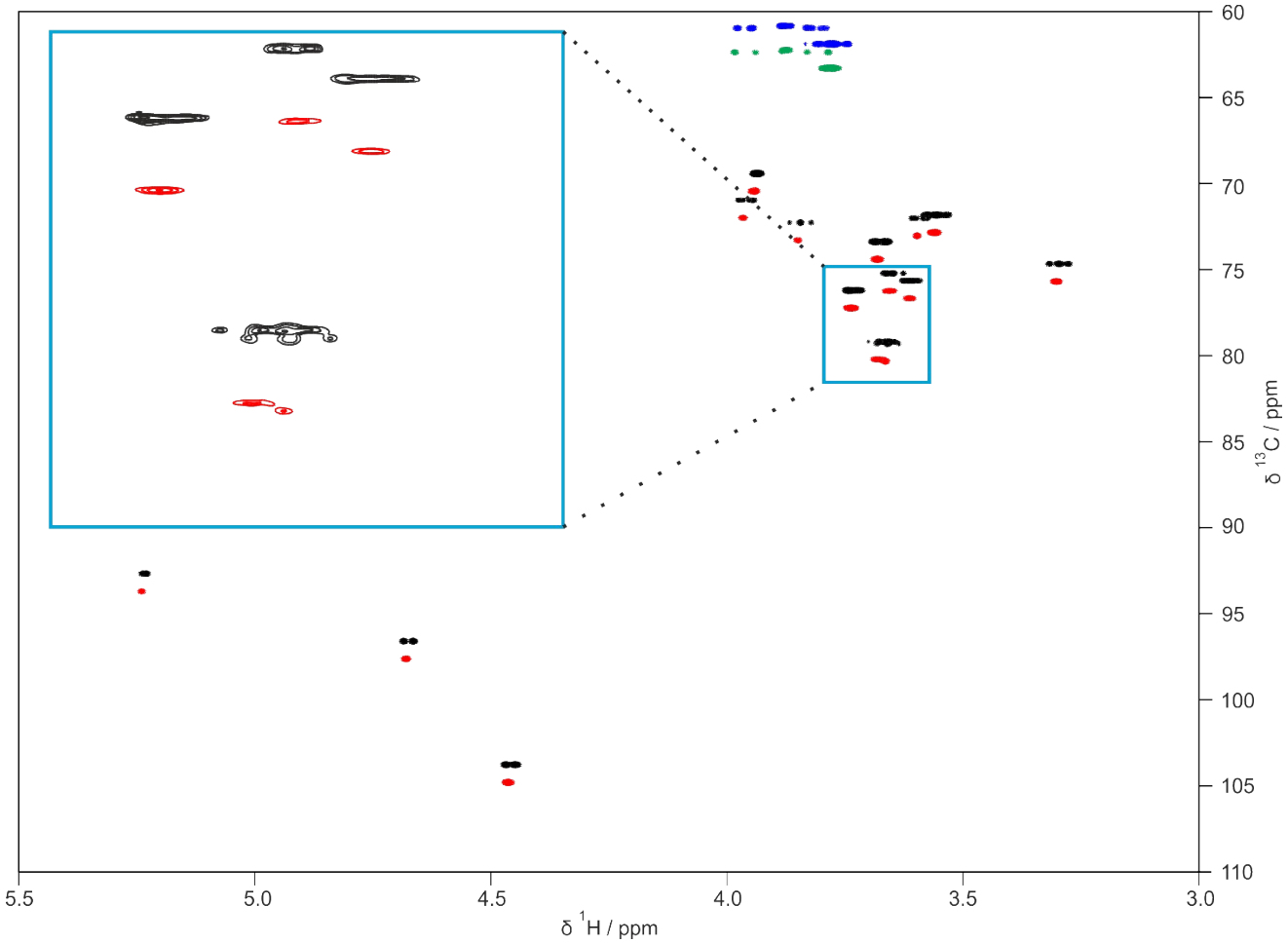

Figure S3. 400 MHz ^{1}H-^{13}C multiplicity-edited HSQC spectra of 450 mM lactose in D_2O with TSP added as an internal reference. Conventional multiplicity-edited HSQC contours are shown in black and blue and the real-time pure shift multiplicity-edited HSQC contours are shown in red and green, shifted vertically in F_1 by 1 ppm for clarity. The inset shows an expansion of the region of the spectrum defined by the blue box.

2. Lacto-*N*-difucohexaose I experimental data

Figure S4 shows a comparison between (a) conventional TOCSY and (b) F_1-PSYCHE-TOCSY data for a sample of 10.2 mg of LNDFH I dissolved in 400 μL D_2O (25 mM) with TSP added as a reference. (c) Covariance processing, as described in section A2, achieves homonuclear decoupling in both dimensions of the spectrum. There is a clear resolution increase in the covariance processed spectrum (c) compared to the conventional spectrum (a). Although covariance can be a very helpful tool, care should be taken when signals overlap as spurious peaks can occur.

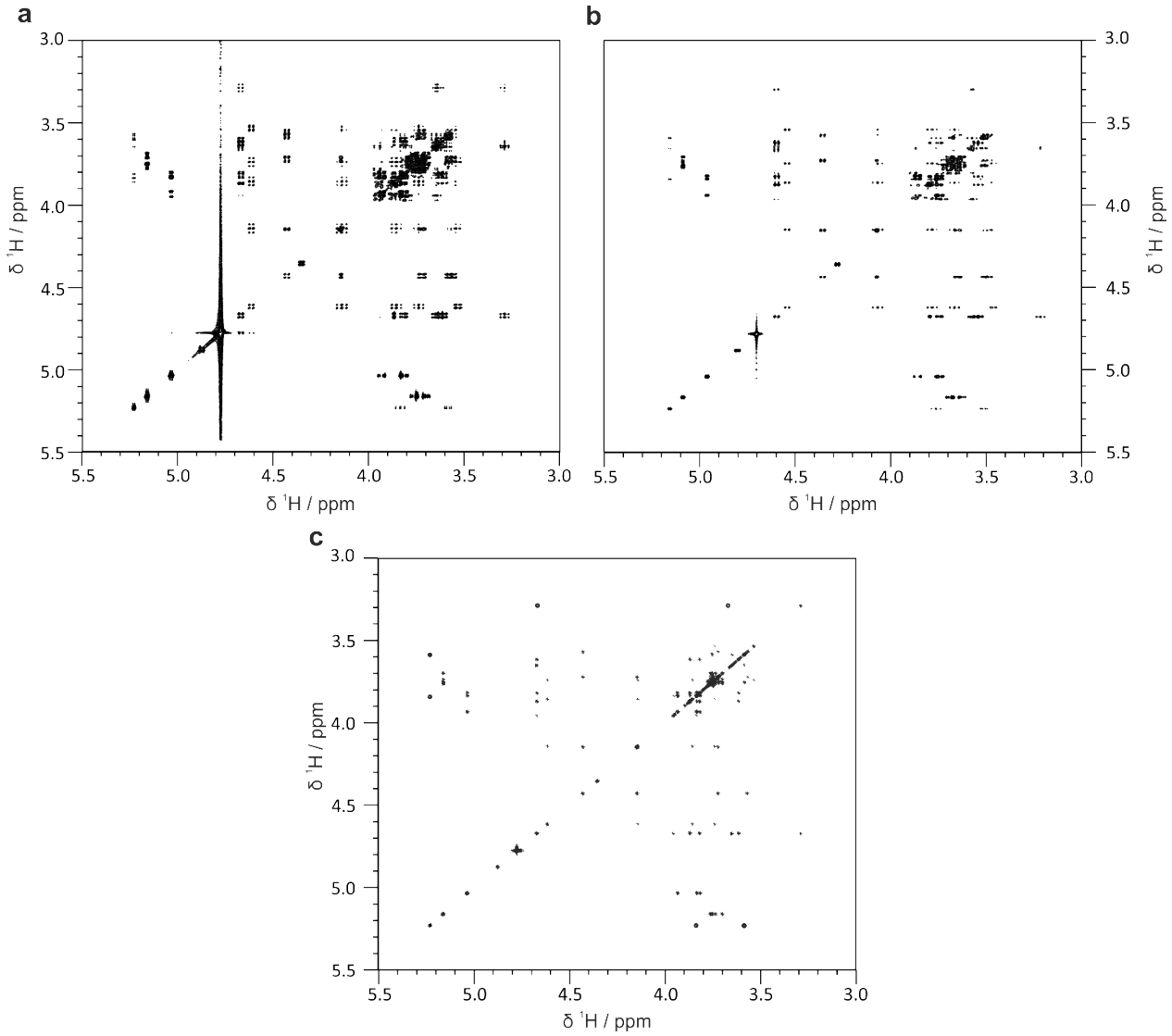

Figure S4. 400 MHz ^{1}H-^{1}H TOCSY spectra of 25 mM lacto-*N*-difucohexaose I in D_2O with TSP added. (a) conventional, (b) F_1-PSYCHE-TOCSY without covariance processing, and (c) F_1-PSYCHE-TOCSY with covariance processing.

Figure S5 shows a comparison between the edHSQC and the edPSHSQC spectrum for a 25 mM sample of LNDFH I in D_2O with TSP added as a reference. The black and blue contours represent the conventional multiplicity-edited HSQC and the red and green contours represent the pure shift multiplicity-edited HSQC (offset vertically by 1 ppm). The methyl groups of LNDFH I at 1.2 ppm have been aliased into the spectrum to maintain resolution in F_1. The inset shows an expansion, defined by the blue box in the spectrum, showing the increase in resolution in the pure shift experiment.

Figure S6 shows a comparison between (a) ^{1}H and (b) ^{13}C literature assignments and the CASPER assignments for LNDFH I. The almost complete correlation between the two datasets shows that the chemical shifts generated by CASPER are in close agreement with those in the literature.

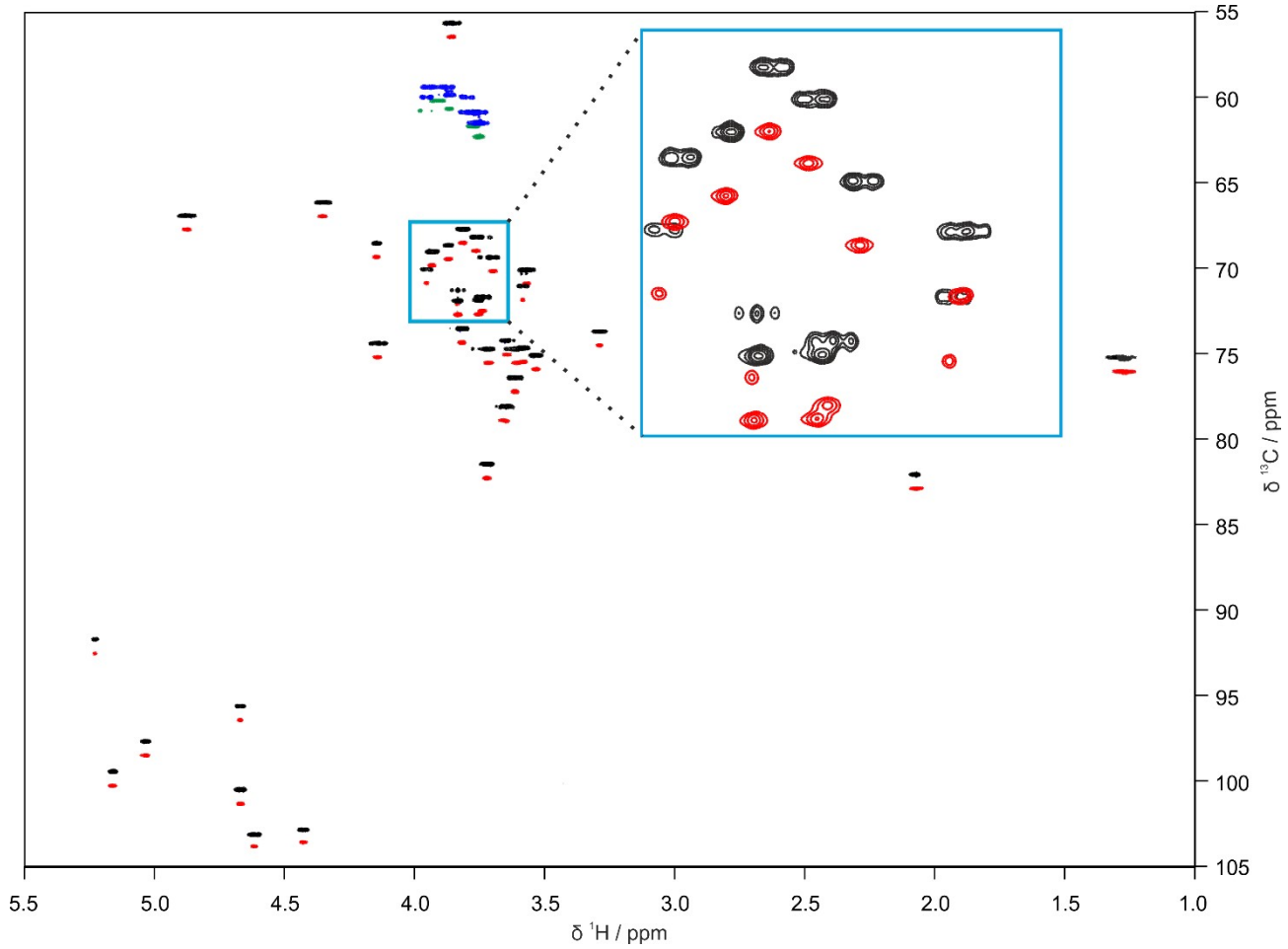

Figure S5. 400 MHz ^{1}H-^{13}C multiplicity edited HSQC spectra of 25 mM lacto-*N*-difucohexaose I in D_2O with TSP added as an internal reference. Conventional multiplicity-edited HSQC contours are shown in black and blue and the real-time pure shift multiplicity-edited HSQC contours are shown in red and green. The pure shift HSQC contours were shifted vertically in F_1 by 1 ppm for clarity. The inset shows an expansion of the region of the spectrum defined by the blue box.

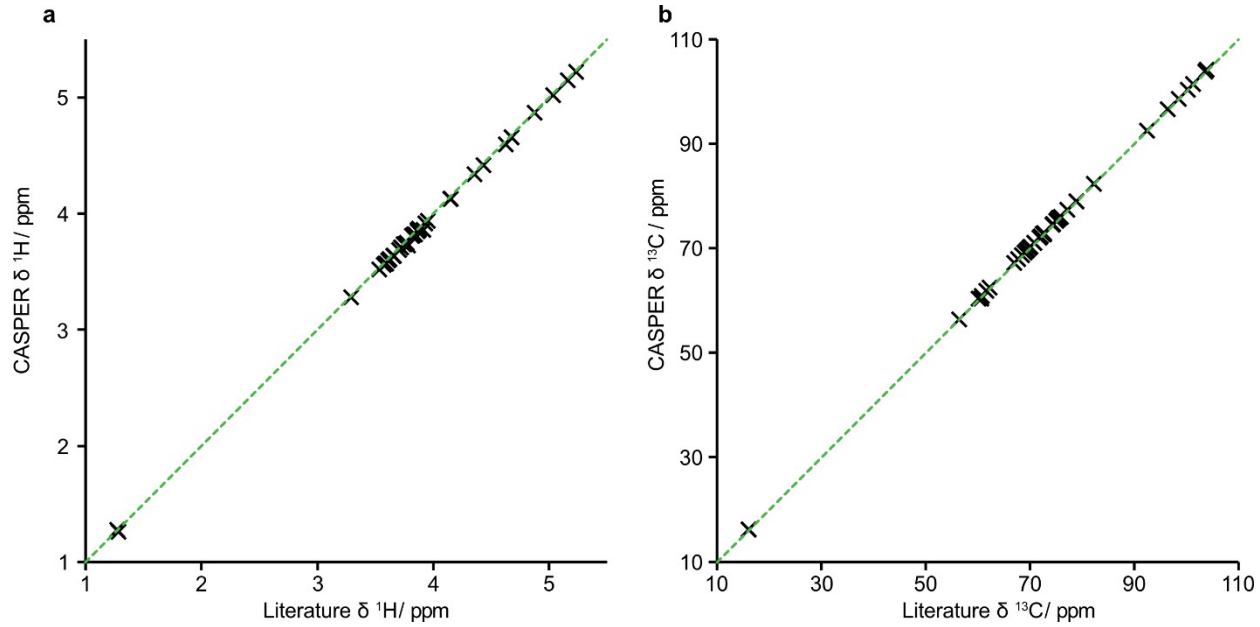

Figure S6. Comparison between the chemical shifts, (a) ^{1}H and (b) ^{13}C, of LNDFH I reported in the literature and generated by CASPER. An appropriate displacement was applied to the ^{13}C NMR chemical shifts to account for the difference in sample temperature between the two datasets.

C. Peak-picking macros

All macros can be downloaded directly from the data repository (DOI:10.48420/22270126). Alternatively, the macros may be downloaded from https://nmr.chemistry.manchester.ac.uk.

1. Real-time BIRD HSQC peak-picking Topspin macro

```
/*UoM_PSHSQC_pp

Developed by Manchester NMR methodology group to peak pick multiplicity edited BIRD Real time
pure shift HSQC data (PSHSQC)
Slight differences in 13C chemical shifts of CH2 (negative) peaks are merged if below a predefined
threshold
CH2 peaks that are doublets in the PSHSQC spectrum are merged if the JHH value is less than a
predefined threshold
13C chemical shifts of aliased peaks can be recalculated by the user inputting the chemical shift of
the peak that was aliased
A new peak list with a user defined name will be generated as a text file

Author: Marshall Smith
Email: marshall.smith@manchester.ac.uk

The peak picking is based on standard Bruker parameters:
MI - Minimum intensity to peak
MAXI - Maximum intensity to pick
F1P - Left limit (max ppm) for peak picking region
F2P - Right limit (min ppm) for peak picking region

PSIGN - pos. (positive peaks only)
        - neg. (negative peaks only)
        - both (positive and negative peaks) - recommended
*/

#define PSIGN_POS      0      /* peak pick positive peaks */
#define PSIGN_NEG      1      /* peak pick negative peaks */
#define PSIGN_BOTH     2      /* peak pick positive and negative peaks*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

   char line[4096],sourcefile[PATH_MAX], targetfile[PATH_MAX], targetfile_2[PATH_MAX],
   targetfile_3[PATH_MAX];
   int min = 0;  // Avoid compilation warnings (may be used uninitialized)
   int max = 0;  // Ditto
   int sum = 0;
   int count = 0;
   int psign;
   double psign1 = 2,download=0;
   int irow,i,j,total=0, total_2=0,cont_2=0, cont=0, ret=1,ret_2=1,n;
```

```
   char str[256],str2[256],str3[256],str4[256],str5[256],str6[256];
   float x[256],xx,y[256],z[256],ppm2[256],ppm1[256],intens[256];
   float array[256], array_2[256],alias_down,alias_up,alias=0;
   float pp1_new[256],pp_new[256],merge_ppm,test[256];
   float pp1[256],pp[256],ppi[256],a,b,c,number;
   float   offs1, offs2, phc0, phc1, thresh=0.04, merge=20;
   double  f1p2, f2p2, f1p1, f2p1, sw2, sw1, sf2 ,sf1, sum1, sum2;
   const char* out="PS_hsqc_peaks";
   char *alias_ppm="0";
   char *alias_ppm_2="0";

// use acquisition parameters for peak picking limits
   FETCHPAR("F1P", &f1p2)
   FETCHPAR("F2P", &f2p2)
   FETCHPARS("OFFSET",&offs2)
   FETCHPARS("SW_p",&sw2)
   FETCHPARS("SF",&sf2)
   sum2 = offs2 - sw2 / sf2;
   FETCHPAR1("F1P", &f1p1)
   FETCHPAR1("F2P", &f2p1)
   FETCHPAR1S("OFFSET",&offs1)
   FETCHPAR1S("SW_p",&sw1)
   FETCHPAR1S("SF",&sf1)
   GETFLOAT("Do you want to pick positive (0), negative (1) or both (2)?",psign1)
   psign=psign1;
   STOREPARS("PSIGN",psign)
   STOREPAR("PSIGN",psign)
   sum1 = offs1 - sw1 / sf1;

        PP2D // peak pick 2D
        ERRORABORT;

   irow=readPeakList(PROCPATH(0));
    if (irow < 0)
   {
      Proc_err(DEF_ERR_OPT, "%s, peak picking aborted", getPeakError());
      ABORT;
   }
   // convert peak list to text file peaks.txt
   XCMD("sendgui convertpeaklist txt");
   ERRORABORT;

   sprintf(sourcefile, "%s/peak.txt", PROCPATH(0)); // file path for peaklist

   //User defined threshold
   GETFLOAT("Input threshold for merging F1 chemical shifts e.g 0.04 ppm",thresh);
   GETFLOAT("Input threshold for CH2 JHH  e.g 20 Hz",merge);

   // User defined naming of editted peak lsit
   GETSTRING("Input desired name for outputted HSQC peak list",out)
                  sprintf(targetfile, "%s/%s.txt", PROCPATH(0),out); // Out put file for edited peaklist
```

```
  FILE *pk = fopen(sourcefile, "r");
  FILE* out_file;
 out_file=     fopen(targetfile,"wt");

  if (pk == NULL)
   {
     fprintf(stderr, "Failed to open file %s for reading\n", sourcefile);
     return 1;
    }

  for (i = 0; i < irow;i++)
  {
  while (fgets(line, sizeof(line), pk) != NULL)
   {
     if (line[0]=='#') continue; /*ignore hashtags in peak.txt*/
     if (isspace(line[3])==0) continue; /*ignore white space*/
     if (line[5]=='#') continue; /*ignore hashtags in field header*/

     sscanf(line,"%s %s %s %s %s %s \n",str,str2,str3,str4,str5,str6); /*read in values from file*/

    ppm2[i]=atof (str4);
    ppm1[i]=atof (str5);
    intens[i]=atof (str6);
    fprintf(out_file, "%.2f %.2f %.2f \n",ppm2[i],ppm1[i],intens[i]); /*print values into new file*/
   }
  }
    fclose(pk);
    fclose(out_file);

   //read temp file we just made to assign values
  out_file=     fopen(targetfile,"r");
  for (i = 0; i < irow;i++)
  {
  fscanf(out_file,"%f %f %f \n",&pp[i],&pp1[i],&ppi[i]); //pp 1H; pp1 13C ; ppi intensity
  }
   fclose(out_file);

   // sort F1 chemical shift in ascending order
for (i = 0; i < irow; ++i)
     {
        for (j = i + 1; j < irow; ++j)
        {
           if (pp1[i] > pp1[j])
           {
              a =  pp[i]; // sort the 1H
              pp[i] = pp[j];
              pp[j] = a;

               b =  pp1[i]; // sorts the 13C
              pp1[i] = pp1[j];
```

```
          pp1[j] = b;

          c =  ppi[i]; // sort intensity
          ppi[i] = ppi[j];
          ppi[j] = c;
         }}}

  for (i = 0; i < irow; ++i)
{
   for (j = i + 1; j < irow; ++j)
   {
   if         (ppi[i]<0 && ppi[j]<0)
   {

                        if (fabs(pp1[i]-pp1[j])<=thresh)
                        {
      if (pp[i] > pp[j])
      {
         a =  pp[i]; // sort the 1H
         pp[i] = pp[j];
         pp[j] = a;

         b =  pp1[i]; // sorts the 13C
         pp1[i] = pp1[j];
         pp1[j] = b;

         c =  ppi[i]; // sort intensity
         ppi[i] = ppi[j];
         ppi[j] = c;
      }
      }
   }}}
 // test whether negative and within threshold

   merge_ppm=merge/sf2;

 for (i = 0; i < irow; ++i)
{
   for (j = i + 1; j < irow; ++j)
   {
   if ((ppi[i]<0 && ppi[j]<0) && (fabs(pp1[i]-pp1[j])>=0) && (fabs(pp1[i] - pp1[j]) <= thresh))

      {
      pp1_new[i]=((pp1[i]+pp1[j])/2);
      pp1[i]=pp1_new[i];
      pp1[j]=pp1_new[i];
      }
  }}

for (i = 0; i < irow; ++i)
 {
```

```
        for (j = i + 1; j < irow; ++j)
        {
        if (ppi[i]<0 && ppi[j]<0)

           if (pp1[j] - pp1[i] <= thresh)
           {
         if (fabs(pp[i] - pp[j])>= 0 && fabs(pp[i] - pp[j]) <= merge_ppm)

           {
           pp_new[i]=((pp[i]+pp[j])/2);

           pp[i]=pp_new[i];
           pp[j]=pp_new[i];
           }
           }}}


for (i = 0; i < irow; ++i)
{
  if (ppi[i]<0 && ppi[i+1]<0 && ppi[i+2]<0 && ppi[i+3]<0)
  {
     if (fabs(pp1[i]-pp1[i+1])<=thresh)
     {
     pp1[i+1]=pp1[i];
     }
     if (fabs(pp1[i]-pp1[i+2])<=thresh)
     {
        pp1[i+2]=pp1[i];
     }
     if (fabs(pp1[i]-pp1[i+3])<=thresh)
     {
        pp1[i+2]=pp1[i];
     }
  }
}


   /*Check if spectral aliasing was performed and calculate the chemical shift of the signal*/
   GETFLOAT("Was spectral aliasing in F1 used in this experiment? \n no(0) \n yes (1)",alias);
    if (alias==1)
    {
       GETFLOAT("Are any peaks aliased down into spectrum? \n no(0) \n yes(1)",alias_down)
       if (alias_down)
       {
    // GETFLOAT("what is the 13C chemical shift of the peak for unfolding?",alias_ppm )
    GETSTRING("what is the 13C chemical shift of the aliased peaks?",alias_ppm );

    while(ret == 1 && total < 256)
       {
     ret = sscanf(alias_ppm, "%f%n", &array[total++], &cont);
     alias_ppm += cont;
```

```
   }
   total--;

        xx=0;
        for (i=0;i<irow;i++)
        {
                for (n=0; n<total; n++)
                {
                if (pp1[i]==array[n])
                {
                        xx=1;
                        pp1[i]=pp1[i]-(sw1/sf1);
                }
        }
                }
                }
                GETFLOAT("Are any peaks aliased up into spectrum? \n no(0) \n yes(1)",alias_up)
                if (alias_up==1)
                {
                        GETSTRING("what is the 13C chemical shift of the aliased
peaks?",alias_ppm_2 );
                        while(ret_2 == 1 && total_2 < 256)
        {
      ret_2 = sscanf(alias_ppm_2, "%f%n", &array_2[total_2++], &cont_2);
      alias_ppm_2 += cont_2;
   }
   total_2--;
        xx=0;
        for (i=0;i<irow;i++)
        {
                for (n=0; n<total; n++)
                {
                if (pp1[i]==array_2[n])
                {
                        xx=1;
                        pp1[i]=pp1[i]+(sw1/sf1);
        }}}}}

    //Generate the CASPER compatible peak list removing any values which are the same
    out_file=   fopen(targetfile,"wt");

   for (i = 0; i < irow;i++)
   {
   if (pp1[i] == pp1[i+1] && pp[i]==pp[i+1] && ppi[i]<0 && ppi[i+1]<0 ) /*if 13C and 1H same and neg
remove*/
            {
            fprintf(out_file,"");
            continue;
            }
   if (pp1[i] - pp1[i+1]<thresh && pp[i]==pp[i+1] && ppi[i]<0 && ppi[i+1]<0 ) /*if 13C and 1H same
and neg remove*/
```

```
            {
            fprintf(out_file,"");
            }
   if ((fabs(pp1[i+1] - pp1[i]<thresh)) && pp[i]==pp[i+1] && ppi[i]<0 && ppi[i+1]<0 ) /*if 13C and 1H
same and neg remove*/
            {
            fprintf(out_file,"");
            }
   if (pp1[i] != pp1[i+1] && pp[i]!=pp[i+1] && ppi[i]<0) /*if 13C different and 1H different and neg
print */
            {
            fprintf(out_file,"%.2f   %.2f \n",pp1[i],pp[i]);
            }

  if (pp1[i] == pp1[i+1] && pp[i]!=pp[i+1] && ppi[i]<0) /*if 13C is the same and 1H different and neg
print */
            {
            fprintf(out_file,"%.2f   %.2f \n",pp1[i],pp[i]);
            }
  if (pp1[i] != pp1[i+1] && pp[i]==pp[i+1] && ppi[i]<0 && pp1[i] - pp1[i+1]>thresh) /*if 13C is
different and 1H is same and neg print */
            {
            fprintf(out_file,"%.2f   %.2f \n",pp1[i],pp[i]);
            }
  if (ppi[i]>0) /*Anything positive is printed*/
            {
            fprintf(out_file,"%.2f   %.2f \n",pp1[i],pp[i]);
            }
    }

    fclose(out_file);

    char flist[PATH_MAX];

getParfileDirForWrite(out, PEAKLIST_DIRS, flist);

dircp(targetfile,flist);

// Read in the generated file to separate the 1H and 13C chemical shifts

int lines=0,ch=0;


FILE *fp2;
fp2=fopen(targetfile,"r");
/* abort if file not found*/
if (fp2 == NULL)
    {
     fprintf(stderr, "Failed to open file %s/%s for reading\n", targetfile);
    return 1;
  }
```

```
 //lines++;

while(!feof(fp2))
{
  ch = fgetc(fp2);
  if(ch == '\n')
  {
    lines++;
  }
}

fclose(fp2);

float pp_hq1[lines],pp_hq2[lines];
// Lets read the values from the HSQC peak list
FILE *fp1;

fp1=fopen(targetfile,"r");

for (i=0; i< lines; i++)
{

   fscanf(fp1,"%f %f \n",&pp_hq1[i],&pp_hq2[i]);  /*pp_hq2 is 1H chemical shift*/
}
fclose(fp1);


sprintf(targetfile_2, "%s/13C_%s.txt", PROCPATH(0),out);
sprintf(targetfile_3, "%s/1H_%s.txt", PROCPATH(0),out);

FILE *out_file2=fopen(targetfile_2,"wt");

for (i = 0; i < lines;i++)
{
if (pp_hq1[i] != pp_hq1[i+1])
{
   fprintf(out_file2,"%.2f \n",pp_hq1[i]);
}

}

fclose(out_file2);
free(out_file2);

FILE *out_file3=fopen(targetfile_3,"wt");

for (i = 0; i < lines;i++)
{
for (j = i + 1; j < lines; ++j)
{
```

```
        if (pp_hq2[i] > pp_hq2[j])
                {
                   a =  pp_hq2[i]; // sort the 1H
                   pp_hq2[i] = pp_hq2[j];
                   pp_hq2[j] = a;
        }}}

        for (i = 0; i < lines;i++)
        {
          fprintf(out_file3,"%.2f \n",pp_hq2[i]);
        }

        fclose(out_file3);
        free(out_file3);

          char out_message[PATH_MAX];
          sprintf(out_message,"Edited HSQC peak list saved to: %s/%s \n  \n Separate 1H and 13C chemical
        shifts are also provided in: 1H_%s and 13C_%s",PROCPATH(0),out,out,out);
QUITMSG(out_message);
```

2. F_1-PSYCHE-TOCSY peak-picking macro

/*UoM_2D_PSTOCSY_pp
Developed by Manchester NMR methodology group to peak pick F1-PSYCHE-TOCSY data
The peak list is compared to the PSHSQC list to remove any correlations that correspond to strong coupling artefact in F1

A new peak list with a user defined name will be generated as a text file

The macro assumes that the edPSHSQC and the F1-PSYCHE-TOCSY are collected under the same folder with different expono

If no values are exported check the referencing as the macro removes anything that is not identical to the edPSHSQC master values

Author: Marshall Smith
Email: marshall.smith@manchester.ac.uk

The peak picking is based on standard Bruker parameters:
MI - Minimum intensity to peak
MAXI - Maximum intensity to pick
F1P - Left limit (max ppm) for peak picking region
F2P - Right limit (min ppm) for peak picking region

PSIGN - pos. (positive peaks only)
- neg. (negative peaks only)
- both (positive and negative peaks) - recommended
*/

```
#define PSIGN_POS      0      /* search for positive peaks */
#define PSIGN_NEG      1      /* search for negative peaks */
#define PSIGN_BOTH     2

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

char   sourcefile[256], targetfile[256];

   char line[4096];
   int min = 0;  // Avoid compilation warnings (may be used uninitialized)
   int max = 0;  // Ditto
   int sum = 0;
   int irow,irow2,i,count,p,l,n,ch=0,lines=0,expno1,j,k,t;
   char str[256],str2[256],str3[256],str4[256],str5[256],str6[256],out[256];
   char PL[256], sourcefile2[PATH_MAX], targetfile2[PATH_MAX], test2[16], targetfile3[PATH_MAX];
   const char* test="PS_hsqc_peaks";
   const char* out2="PS_tocsy_peaks";
   float x,y[irow],z[irow],ppm2[irow],ppm1[irow],intens[irow],pp1[512],pp[512],ppi[512];
```

```
float   a,b,c,offs1, offs2, phc0, phc1, merge, thresh, thresh_max,pp_r[4096];
float pp_hq1[512],pp_hq2[512];
double  f1p2, f2p2, f1p1, f2p1, sw2, sw1, sf2 ,sf1, sum1, sum2,download=0;
static int psign=0;
count=0;
thresh=90;
thresh_max=115;

FETCHPAR("F1P", &f1p2)
FETCHPAR("F2P", &f2p2)
FETCHPARS("OFFSET",&offs2)
FETCHPARS("SW_p",&sw2)
FETCHPARS("SF",&sf2)
sum2 = offs2 - sw2 / sf2;
FETCHPAR1("F1P", &f1p1)
FETCHPAR1("F2P", &f2p1)
FETCHPAR1S("OFFSET",&offs1)
FETCHPAR1S("SW_p",&sw1)
FETCHPAR1S("SF",&sf1)
FETCHPARS("PSIGN", &psign)
sum1 = offs1 - sw1 / sf1;

     PP2D // peak pick 2D
     ERRORABORT;

irow=readPeakList(PROCPATH(0));

 if (irow < 0)
{
   Proc_err(DEF_ERR_OPT, "%s, peak picking aborted", getPeakError());
   ABORT;
}
// convert peak list to text file peaks.txt
XCMD("sendgui convertpeaklist txt");
ERRORABORT;

sprintf(sourcefile, "%s/peak.txt", PROCPATH(0)); // file path for peak list

// User defined naming of edited peak lsit
     sprintf(targetfile, "%s/temp", PROCPATH(0)); // Out put file for edited peak list
FILE *pk = fopen(sourcefile, "r");
FILE* out_file;
out_file=     fopen(targetfile,"wt");

if (pk == NULL)
  {
   fprintf(stderr, "Failed to open file %s for reading\n", sourcefile);
   return 1;
  }

for (i = 0; i < irow;i++)
```

```
{
while (fgets(line, sizeof(line), pk) != NULL)
 {
   if (line[0]=='#') continue; /*ignore hashtags in peak.txt*/
   if (isspace(line[2])==0) continue; /*ignore white space*/
   if (line[5]=='#') continue; /*ignore hashtags in field header*/

   sscanf(line,"%s %s %s %s %s %s \n",str,str2,str3,str4,str5,str6); /*read in values from file*/
  ppm2[i]=atof (str4);
  ppm1[i]=atof (str5);
  intens[i]=atof (str6);
  fprintf(out_file, "%.2f %.2f %.2f \n",ppm2[i],ppm1[i],intens[i]); /*print values into new file*/
 }
}
  fclose(pk);
  fclose(out_file);


 //read temp file we just made to assign values
out_file=    fopen(targetfile,"r");
for (i = 0; i < irow;i++)
{
fscanf(out_file,"%f %f %f \n",&pp[i],&pp1[i],&ppi[i]);
}
 fclose(out_file);

 // sort F1 chemical shift in ascending order

for (i = 0; i < irow; ++i)
    {
       for (j = i + 1; j < irow; ++j)
       {
          if (pp1[i] < pp1[j])
          {
             a =  pp[i];
             pp[i] = pp[j];
             pp[j] = a;

                                                b =  pp1[i];
             pp1[i] = pp1[j];
             pp1[j] = b;

             c =  ppi[i];
             ppi[i] = ppi[j];
             ppi[j] = c;
          }}}
 out_file=    fopen(targetfile,"wt");

for (i = 0; i < irow;i++)
{
         fprintf(out_file,"%.2f %.2f \n", pp[i],pp1[i]);
```

```
    }
    fprintf(out_file,"%d \n",irow);
    fclose(out_file);


     /*Lets get the HSQC data file*/
 //GETINT("Input expno for PSHSQC data: ",expno1)
 GETSTRING("input edited hsqc peak list name",test);

   char speaklistdir[PATH_MAX];

if (getParfileDirForRead(test, PEAKLIST_DIRS, sourcefile2)<0)
{
   sprintf(speaklistdir, "%s/stan/nmr/lists/peaklist/", PathXWinNMRExp());

    fprintf(stderr, "Failed to open file %s%s for reading\n", speaklistdir,test);
  ABORT
}

FILE *fp;
fp=fopen(sourcefile2,"r");

 if (fp == NULL)
     {
      fprintf(stderr, "Failed to open file %s for reading\n", test);
      return 1;
     }
 lines++;
while(!feof(fp))
{
  ch = fgetc(fp);
  if(ch == '\n')
  {
    lines++;
  }
}
fclose(fp);
free(fp);

// Lets read the values from the HSQC peak list
FILE *fp1;
fp1=fopen(sourcefile2,"r");
for (i=0; i< lines; i++)
{
  fscanf(fp,"%f %f \n",&pp_hq1[i],&pp_hq2[i]);
}
fclose(fp);

// Check the 13 Chemical shift of F1 chemical shifts?
GETFLOAT("Input minimum 13C chemical shift for anomeric region",thresh);
GETFLOAT("Input maximum 13C chemical shift for anomeric region",thresh_max);
```

```
float keep[256];
sprintf(targetfile3, "%s/peaks_removed", PROCPATH(0),out2);
FILE *out_file3;
out_file3=fopen(targetfile3,"w");
k=0;
for (p=0; p<irow; p++)
{
for (l=0; l<lines; l++)

  if (pp1[p] == pp_hq2[l])
   {
   if (pp_hq1[l]>thresh && pp_hq1[l]<thresh_max)
   {
   keep[k]=pp_hq2[l];
   k=k+1;
   }
   continue;
   }
   }

for (i=0; i<lines; i++)
{
   for (j=i+1; j<lines; j++)
   {
      if (pp_hq2[i]==pp_hq2[j])
      {pp_hq2[j]=0;}
   }
}

for (p=0; p<count;p++)
{
        fprintf(out_file3,"removed %f \n",pp_r[p]);
}
fclose(out_file3);

GETSTRING("Input name for edited TOCSY peak list",out2);
sprintf(targetfile2, "%s/%s.txt", PROCPATH(0),out2);
int r;
FILE *out_file2;
out_file2=fopen(targetfile2,"w");

for (l=0; l<lines; l++)
{
for (p=0; p<irow; p++)

{
  if (pp[p] == pp_hq2[l] && pp[p] != pp1[p] && fabs(pp[p]-pp1[p])>0.02 ) // check pp exists in hsqc and not
a diagonal peak
   {
        x=1;
```

```
        for (r=0; r<k; r++)
        {
        if (pp1[p]==keep[r])
        {
           x=0;
        }
        }
        if (x==0)
        {
        fprintf(out_file2,"%.2f %.2f\n",pp[p],pp1[p]);
        }
                }}}


fclose(out_file2);
free(out_file2);

char flist[PATH_MAX];

getParfileDirForWrite(out2, PEAKLIST_DIRS, flist);

dircp(targetfile2,flist);


  char out_message[PATH_MAX];
  sprintf(out_message,"file saved to %s",targetfile2);
QUITMSG(out_message);
```

3. Selective pure shift TOCSY peak-picking macro

```
/*Developed by Manchester NMR methodology group to peak pick 1D pure shift selective TOCSY data

The peak list is compared to the PSHSQC data to automatically remove pure shift artefacts

A new peak list with a user defined name will be generated as a text file

Author: Marshall Smith
Email: marshall.smith@manchester.ac.uk

The peak picking is based on standard Bruker parameters:
MI - Minimum intensity to peak
MAXI - Maximum intensity to pick
F1P - Left limit (max ppm) for peak picking region
F2P - Right limit (min ppm) for peak picking region

PSIGN - pos. (positive peaks only)
                        - neg. (negative peaks only)
                        - both (positive and negative peaks) - recommended
*/

#define PSIGN_POS      0      /* search for positive peaks */
#define PSIGN_NEG      1      /* search for negative peaks */
#define PSIGN_BOTH     2

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

char   sourcefile[256], targetfile[256];

   char line[4096];
   int min = 0;  // Avoid compilation warnings (may be used uninitialized)
   int max = 0;  // Ditto
   int sum = 0;
   int count = 0;
   int irow,i,j,k,p,l,expno1,irow2, ch=0,lines=0, lines2=0, ch2=0;
   char str[256],str2[256],str3[256],str4[256],str5[256],out[256], filename[256];
   char PL[256],sourcefile2[PATH_MAX],targetfile2[256];
   float a,b,c,x,y[irow],z[irow],ppm2[irow],ppm1[irow],intens[irow],ppm22[256],ppm12[256];
   float offs1, phc0, phc1, cnst12;
   float pp1[256],pp[256],ppi[irow],pks[256],pp_hq1[lines],pp_hq2[lines];
   double   f1p1, f2p1, sw1 ,sf1, sum1,download=0;
   const char* test="PS_hsqc_peaks",*out2="PS_tocsy_peaks";
   static int psign=0;
   float thresh=90;      // default minimum 13C chemical sift for anomeric signal
   float thresh_max=115;  // default maximum 13C chemical shift for anomeric signal
```

```
// use acquisition parameters for peak picking limits
   FETCHPAR("F1P", &f1p1)
   FETCHPAR("F2P", &f2p1)
   FETCHPAR("OFFSET",&offs1)
   FETCHPAR("SW_p",&sw1)
   FETCHPAR("SF",&sf1)
   FETCHPAR("PSIGN", &psign)
  // FETCHPAR("cnst12",&cnst12)
   sum1 = offs1 - sw1 / sf1;

        PP // peak pick 1D

        ERRORABORT;

   irow=readPeakList(PROCPATH(0));

    if (irow < 0) /*if no peaks abort*/
   {
      Proc_err(DEF_ERR_OPT, "%s, peak picking aborted", getPeakError());
      ABORT;
   }
   // convert peak list to text file peaks.txt
   XCMD("sendgui convertpeaklist txt");
   ERRORABORT;

   sprintf(sourcefile, "%s/peak.txt", PROCPATH(0)); // file path for peak list
        sprintf(targetfile, "%s/temp", PROCPATH(0)); // temp out put file for edited peak list

   FILE *pk = fopen(sourcefile, "r");
   FILE* out_file;
   out_file=     fopen(targetfile,"wt");

   if (pk == NULL)
     {
      fprintf(stderr, "Failed to open file %s for reading\n", filename);
     return 1;
   }

   /*parse the 1D peak list and read values*/
   for (i = 0; i < irow;i++)
   {
   while (fgets(line, sizeof(line), pk) != NULL)
    {
      if (line[0]=='#') continue; /*ignore hashtags in peak.txt*/
      if (isspace(line[2])==0) continue; /*ignore white space*/
      if (line[11]=='#') continue; /*ignore hashtags in field header*/
      if (line[28]=='H') continue; /*ignore hashtags in field header*/

      sscanf(line,"%s %s %s %s %s \n",str,str2,str3,str4,str5); /*read in values from file*/

     ppm1[i]=atof (str4);
     intens[i]=atof (str5);
    fprintf(out_file, "%.2f %.2f \n",ppm1[i],intens[i]); /*print values into new file*/
     }
   }
```

```
    fclose(pk);
     fclose(out_file);

   //read temp file we just made to assign values
   out_file=     fopen(targetfile,"r");

   for (i = 0; i < irow;i++)
   {
   fscanf(out_file,"%f %f \n",&pp[i],&ppi[i]);
   }
    fclose(out_file);
    // sort 1H TOCSY chemical shifts in ascending order


 for (i = 0; i < irow; ++i)
     {
        for (j = i + 1; j < irow; ++j)
        {
           if (pp[i] > pp[j])
           {
              a =  pp[i];
              pp[i] = pp[j];
              pp[j] = a;

              c =  ppi[i];
              ppi[i] = ppi[j];
              ppi[j] = c;
           }
        }
     }

    out_file=    fopen(targetfile,"wt");
   fprintf(out_file,"F1(ppm) \n");
   for (i = 0; i < irow;i++)
   {

           fprintf(out_file,"%.2f \n", pp[i]);

    }
    fclose(out_file);

     /*Lets get the HSQC data file*/
 //GETINT("Input expno for PSHSQC data: ",expno1)
 GETSTRING("input edited hsqc peak list name",test);

  //sprintf(sourcefile2, "%s/%s/%d/pdata/%d/%s", disk, name, expno1,procno,test);

  char speaklistdir[PATH_MAX];

if (getParfileDirForRead(test, PEAKLIST_DIRS, sourcefile2)<0)
{
   sprintf(speaklistdir, "%s/stan/nmr/lists/peaklist/", PathXWinNMRExp());

    fprintf(stderr, "Failed to open file %s%s for reading\n", speaklistdir,test);
```

```
 ABORT
}



 //
FILE *fp;
fp=fopen(sourcefile2,"r");
/* abort if file not found*/
if (fp == NULL)
     {
       fprintf(stderr, "Failed to open file %s/%s for reading\n", speaklistdir,test);
      return 1;
   }

 lines++;

while(!feof(fp))
{
  ch = fgetc(fp);
  if(ch == '\n')
  {
    lines++;
  }
}

fclose(fp);
free(fp);

// Lets read the values from the HSQC peak list
FILE *fp1;

fp1=fopen(sourcefile2,"r");

for (i=0; i< lines; i++)
{

   fscanf(fp,"%f %f \n",&pp_hq1[i],&pp_hq2[i]);  /*pp_hq2 is 1H chemical shift*/
}
fclose(fp);
/*get user defined limits for 13C limits for anomeric signal*/
    GETFLOAT("Input minimum 13C chemical shift for anomeric region",thresh);
    GETFLOAT("Input maximum 13C chemical shift for anomeric region",thresh_max);

/*find the anomeric signal based on 13C chemical shift*/
for (p=0; p<irow; p++)
{
for (l=0; l<lines; l++)

   if (pp[p] == pp_hq2[l])
    {
    if (pp_hq1[l]>thresh && pp_hq1[l]<thresh_max)
    {
    cnst12=pp_hq2[l];
    }
```

```
   continue;
   }
   }

   if (cnst12==0)
    {
      fprintf(stderr, "Failed to find anomeric signal. Check referencing to HSQC\n");
     return 1;
   }

GETSTRING("Input name for edited tocsy peak list",out2);
sprintf(targetfile2, "%s/%s.txt", PROCPATH(0),out2);

FILE *out_file2;
out_file2=fopen(targetfile2,"w");

for (p=0; p<irow; p++)
{
for (l=0; l<lines; l++)

  if (pp[p] == pp_hq2[l])
   {

   fprintf(out_file2,"%.2f \n",pp[p]);

   continue;
   }
   }

fclose(out_file2);

FILE *fp2;
fp2=fopen(targetfile2,"r+");
/* abort if file not found*/
if (fp2 == NULL)
    {
      fprintf(stderr, "Failed to open file %s for reading\n", targetfile2);
     return 1;
   }
 //lines2++;
while(!feof(fp2))
{
 ch2 = fgetc(fp2);
 if(ch2 == '\n')
 {
  lines2++;
 }
}
/*check the output contains some data*/
if (lines2 == 0)
    {
      fprintf(stderr, "The file %s contains no peaks check referencing of PSHSQC and TOCSY data", targetfile2);
     return 1;
   }
fclose(fp2);
```

```
/*Remove any accidental repeats due to overlap in the HSQC*/
fp2 =  fopen(targetfile2,"r");

   for (i = 0; i < lines2;i++)
   {
   fscanf(fp2,"%f \n",&pks[i]);
   }
    fclose(fp2);

 fp2 = fopen(targetfile2,"w");

   for (i = 0; i < lines2;i++)
   {
        if (pks[i] != pks[i+1] && pks[i]!=cnst12)
        {
   fprintf(fp2,"%.2f %.2f \n",cnst12,pks[i]);
        }
   }
   fclose(fp2);

       char flist[PATH_MAX];

getParfileDirForWrite(out2, PEAKLIST_DIRS, flist);

dircp(targetfile2,flist);


  char out_message[PATH_MAX];
  sprintf(out_message,"file saved to %s",targetfile2);
QUITMSG(out_message);
```