J Biomol NMR (2012) 52:79–89
DOI 10.1007/s10858-011-9584-3

ARTICLE

Al NMR: a novel NMR data processing program optimized for sparse sampling

John M. Gledhill Jr · A. Joshua Wand

Received: 21 August 2011 / Accepted: 27 October 2011 / Published online: 15 November 2011

Abstract Sparse sampling in biomolecular multidimensional NMR offers increased acquisition speed and resolution and, if appropriate conditions are met, an increase in sensitivity. Sparse sampling of indirectly detected time domains combined with the direct truly multidimensional Fourier transform has elicited particular attention because of the ability to generate a final spectrum amenable to traditional analysis techniques. A number of sparse sampling schemes have been described including radial sampling, random sampling, concentric sampling and variations thereof. A fundamental feature of these sampling schemes is that the resulting time domain data array is not amenable to traditional Fourier transform based processing and phasing correction techniques. In addition, radial sampling approaches offer a number of advantages and capabilities that are also not accessible using standard NMR processing techniques. These include sensitivity enhancement, sub-matrix processing and determination of minimal sets of sampling angles. Here we describe a new software package (Al NMR) that enables these capabilities in the context of a general NMR data processing environment.

Keywords Sparse sampling · Multidimensional Fourier transform · NMR data processing

Introduction

Sparse sampling in biomolecular multidimensional NMR offers increased acquisition speed and resolution (Eghbalnia et al. 2005; Hiller et al. 2005; Jaravine et al. 2006; Kim and Szyperski 2003; Kupce and Freeman 2004; Stern et al. 2002; Szyperski et al. 1993) and, if appropriate conditions are met, an increase in sensitivity (Gledhill and Wand 2010). Here we are most interested in the features of sparse systematic sampling of indirectly detected chemical shift evolution. The coupling of incremented time domains in solution NMR spectroscopy finds its roots in the "accordion" class of multidimensional NMR experiments where a mixing period (τ_m) and a chemical shift evolution period (t_1) is systematically co-evolved (i.e. $t_1 = \kappa\tau_m$) (Bodenhausen and Ernst 1982). The concept was then applied in the reduced dimensionality experiments to two indirect chemical shift evolution periods where the two chemical shifts are expressed as a doublet with one peak arising from their sum and another from their difference (Szyperski et al. 1993). To assure that both doublet components are contained within the selected sweep width a scaling factor is utilized such that $t_1 = \tau$ and $t_2 = \tau\kappa$. To further generalize the reduced dimensionality technique, Brutscher and coworkers determined how to isolate the individual quadrature components of the doublet (Brutscher et al. 1995). The subsequent GFT approach generalized the peak component isolation to experiments of arbitrary dimensionality

Electronic supplementary material The online version of this article (doi:10.1007/s10858-011-9584-3) contains supplementary material, which is available to authorized users.

J. M. Gledhill Jr · A. J. Wand (✉)
Graduate Group in Biochemistry and Molecular Biophysics, Perelman School of Medicine, University of Pennsylvania, Philadelphia, PA 19104-6059, USA
e-mail: wand@mail.med.upenn.edu

A. J. Wand
Department of Biochemistry and Biophysics, Perelman School of Medicine, University of Pennsylvania, 905 Stellar-Chance Laboratories, 422 Curie Blvd, Philadelphia, PA 19104-6059, USA

and developed a powerful analytical strategy to obtain the underlying frequency components without the need to engage an intermediate multidimensional NMR frequency spectrum (Kim and Szyperski 2003). In contrast, the projection reconstruction approach sought to generate a complete multidimensional frequency spectrum that is amenable to traditional processing (Freeman and Kupce 2003; Kupce and Freeman 2004). Though similar to reduced dimensionality in that a scaling factor between the two incremented time domains is employed, the increments are scaled systematically via a geometric relationship such that $t_1 = \tau\cos(\alpha)$ and $t_2 = \tau\sin(\alpha)$ where α is termed the sampling angle. Utilizing the full quadrature detection scheme of Brutscher and coworkers (Brutscher et al. 1995), the projection cross section theorem could be used to project the individual sampling angle data sets into a final multidimensional spectrum (Freeman and Kupce 2003; Kupce and Freeman 2003, 2004). It was soon realized, however, that the computational expense of projecting the individual scaling factor data sets could be circumvented by use of a true multidimensional Fourier transform (Kazimierczuk et al. 2006a, b) and that this approach actually accommodates random sampling of the indirect dimensions (Kazimierczuk et al. 2006a, b, 2010a, b). Sparse sampling of incremented times domains of indirectly detected dimensions combined with the direct truly multidimensional Fourier transform (Coggins and Zhou 2006; Kazimierczuk et al. 2006a, b; Marion 2006) has elicited particular attention because of the ability to generate a final spectrum amenable to traditional methods of analysis.

A number of sparse sampling schemes have been utilized in conjunction with the true multidimensional Fourier transform. These include radial sampling, random sampling, concentric sampling and variations thereof (Coggins and Zhou 2007; Coggins and Zhou 2008; Hoch et al. 2008; Kazimierczuk et al. 2006a, b, 2008, 2010a, b; Pannetier et al. 2007). Each sparse sampling scheme has advantages and disadvantages and the reader is referred to comprehensive reviews by Coggins et al. (2010) and Kazimierczuk et al. (2010a, b) for further details. The optimum sampling scheme is often dependent on the desired application, particularly in regard to the spectral artifacts that may result. Sparse sampling often results in an underdetermined data set and the final resulting spectrum contains artifacts. The details of the sampling artifacts are directly dependent on the sampling pattern applied. In general, randomly sampled experiments produce artifacts that are manifested as baseline noise and radial sampled experiments produce coherent ridges that extend from the authentic peak chemical shifts at the defined sampling angles (Kazimierczuk et al. 2010a, b). The incoherence of random sampling artifacts makes this scheme appropriate for most applications as they are largely introduced as white noise. In many situations, however, the deterministic nature of the artifacts and statistical nature of the data makes radial sampling optimal, particularly when prior information about the chemical shifts is available (Gledhill et al. 2009; Gledhill and Wand 2010).

Processing sparsely sampled data is generally a two-step procedure. An overview of processing steps for radially sampled three-dimensional data is shown in Fig. 1. The acquisition dimension, being acquired in the usual time sequential way, is processed using traditional digital filtering, zero-filling and so on followed by fast Fourier transformation and frequency domain phase correction. Quadrature components for the indirectly detected dimensions are appropriately isolated and stored to an intermediate two-dimensional matrix with the four quadrature components of each time point interleaved (Fig. 1b). The indirectly detected sparsely sampled time domains may then be transformed with a true direct two-dimensional Fourier transform (Coggins and Zhou 2006; Kazimierczuk et al. 2006a, b; Marion 2006). This process is repeated for each sampling angle collected, storing each matrix separately. Randomly sampled data is processed in a similar manner except the time points must be supplied to the direct two-dimensional Fourier transform. Various artifact removal algorithms can then be applied to the final matrix to remove or suppress artifacts arising from the particular sampling scheme employed.

Sparsely sampled data requires a non-standard array of processing capabilities and strategies that are not available in commercial or non-commercial processing platforms. Traditional processing programs that are generally available, such as NMRpipe (Delaglio et al. 1995), are capable of processing the acquisition dimension but are quite limited with respect to processing sparsely sampled data, unless the data is appropriately filled with zeros to make it applicable for fast Fourier transform or a numerical processing strategy such as maximum entropy is utilized. This is particularly acute in the context of processing sparsely sampled data in a phase sensitive manner where there are 2^n quadrature components for n-linked time domains. In this context the user would need to store each of the quadrature components in a different matrix location while a matrix amenable to storing an arbitrary number of quadrature components would simplify this procedure. The absence of an appropriate generally available processing platform is particularly limiting for the phase correction of radial data (Gledhill and Wand 2007). In addition, several novel features of sparsely sampled data are completely inaccessible using currently available processing packages. Examples for radial data processing include ridge removal (Kupce and Freeman 2003), statistical sensitivity enhancement (SEnD) (Gledhill and Wand 2010), submatrix processing and optimized intensity extraction

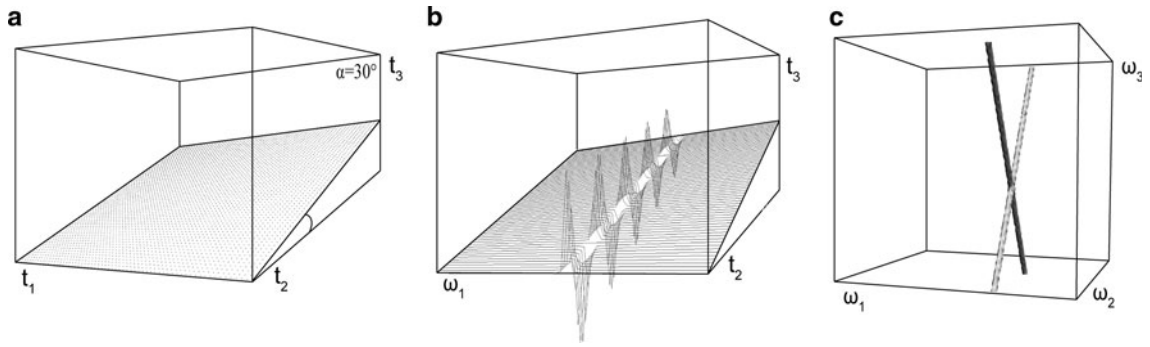

Fig. 1 An overview of the two-step procedure required to process radial sampled data using generated data. **a** depicts radial sampled time domain data that was sampled at $\alpha = 30°$. A plane of data is collected in the three dimensional time domain by Cartesian sampling the acquisition dimension and radial sampling the two indirect dimensions. Radial sampling is achieved by coevolving the two indirect dimensions such that $t_1 = n\cos(\alpha)/sw_1$ and $t_2 = n\sin(\alpha)/sw_2$., where α is the sampling angle, n is the common increment number and sw are the desired sweep widths. Processing the data requires two steps; first, all of the acquisition dimension vectors are processed using common fast Fourier transform methodology and second, the direct multi-dimensional Fourier transform is used to process both indirect dimensions simultaneously. The mixed mode spectrum is shown in **b** where the acquisition dimension has been transformed to the frequency domain while the indirect dimensions are still in the time domain. The final spectrum after application of the direct multi-dimensional Fourier transform to each vector spanning the two indirect dimension time domains is shown in **c**. The resulting frequency domain spectrum demonstrates the ridge artifacts inherent to radial sampling

(Gledhill et al. 2009), and minimum sampling angle set determination (Gledhill and Wand 2008).

To overcome these and other limitations of existing processing packages we have developed a new processing platform, Al NMR. Al NMR is capable of processing both Cartesian and arbitrarily sampled data with the fast Fourier transform and the direct multidimensional Fourier transform, respectively. To enhance user flexibility Al NMR is built around the Python scripting language, allowing users to write custom processing scripts that are amenable to essentially any sampling scheme or acquisition order. Particular attention is paid to radially sampled data with regard to phase correction, optimized angle selection and SEnD optimization. The program features an intuitive analysis interface and a Python command line that facilitates real time processing and analysis.

Program organization

Al NMR is available as a cross platform Python module that provides all command line functionality or as a standalone Windows program that contains a graphical interface complete with a Python command interpreter, Python engine and an interactive spectrum plotting and analysis interface. Both versions have the same processing engine based on the Python scripting language. Figure 2 shows a diagram of the program organization. Al NMR uses the Python scripting language to interact with the user by extending the standard Python language with NMR data processing objects and functions. Basing the program on the Python scripting language presents numerous obvious advantages. The core of the program is the Al NMR

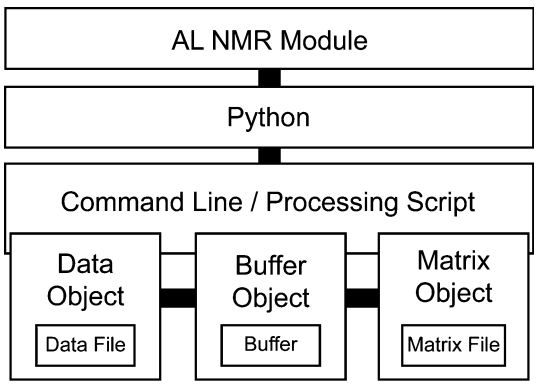

Fig. 2 The Al NMR program organization. The Al NMR Python module is loaded into Python as a dynamic linked C++ library. This module extends the standard capacity of Python to include NMR data processing objects and functions. The user interacts with the module using either the Python command line or through processing scripts executed by the Python interpreter. Typically a processing script will create a data object to interact with the primary data files. FIDs are read using the data object member function to create a buffer object. The buffer object contains the FID data and all of the necessary processing functions. The FID is processed using the buffer object member functions and the 1D spectrum is written into a user created matrix file. The output matrix file is accessed through the associated matrix object member functions

module that contains the objects and functions for NMR data processing. The module is written in C++ to enable advantages of current multi-core architecture and utilizes the FFTW (Frigo and Johnson 2005) and LAPACK (Anderson 1999) libraries for fast Fourier transforms and linear algebra, respectively. The user interacts with the package through command line or processing scripts. Processing scripts are the essential feature of the program. Typically a script will read an NMR data file into a buffer,

then process and store the buffer into a final matrix. These steps are summarized below.

Al NMR python module

Al NMR is based on a custom C++ module loaded into Python. This module extends Python to include a set of NMR data processing objects and functions while retaining the standard Python functionality. The package includes all of the central NMR data processing functions typically employed in the processing of NMR data collected with traditional uniform time intervals and a new set of functions to process arbitrarily sampled NMR data. Al NMR uses an object oriented approach for data processing, which simplifies the syntax of data processing, increases readability and decreases processing time. Object oriented data processing also increases the flexibility of processing scripts. Scripts are adaptable for essentially any type of sampling or acquisition scheme.

Al NMR contains three main objects: an NMR data object which is the users interface with Bruker or Varian spectrometer files, a vector buffer object which provides data processing capacities and a n-dimensional matrix file object. There are also two auxiliary objects that support the main objects; memory matrix objects and output matrix objects. Memory matrix objects serve as an intermediate between data processing and matrix objects stored on a disk. These objects are available in two and three dimensions. The output matrix object allow the user to save matrix in either Sparky or Felix file formats for of these popular graphical analysis packages.

Data object

The NMR data object acts as a direct interface between Al NMR and the primary Varian or Bruker spectrometer data file. When the data object is created the experiment parameters are automatically stored as attributes of the object. Table 1 shows the data object attributes available to the user. The attributes are accessed using the dot operator. For example, the dimension of a Bruker data file is returned from the command `bdat.dim`. All attributes, excluding the spectrum dimension, are returned as Python lists with a value for each dimension. The values for each dimension are accessed using standard Python list indexing. For example, `bdat.td[0]` returns the total number of points in the acquisition dimension. Time domain data is read directly from the data object using the read member function, which returns one free induction decay (FID). Optional keyword values control which FID is read. The function prototype is:

```
fid = dataobj.read(fidnum = 'nextfid', convert
    = True)
```

In this example, the arguments specify the relative FID number and that it will be converted from the Bruker digital format, respectively. Note that the convert option is unique to the Bruker data object and is required to remove the decimation buildup. The data is converted by Fourier transformation, appropriate phase correction, inverse Fourier transformation and removal of trailing zeros.

Buffer object

The buffer object provides memory allocation to hold a FID or one-dimensional spectrum in memory, attributes of the FID or spectrum and functions to process or manipulate the data. There are three methods to create a buffer object; reading a FID from the data object, creating an empty buffer or by reading a vector from a matrix object. There are no limitations to the number of data points a buffer can hold other than available computer memory. There are also no limits to the number of quadrature components in a buffer. Buffer data points are edited using standard Python list indexing. If the number of quadrature components is greater than one each element of the buffer is defined using a list. For example, each element in a buffer containing complex data is set using as a two element list: `buf[0] = [r,i]`

Here r and i represent the real and imaginary components, respectively. Sparsely sampled data is defined similarly, using one value for each quadrature component. For example, a single time point in (3,2) radial sampled data will contain four quadrature components. In this case r and i represent the cosine and sine modulated components with respect to the first or second increment time dimension and the data points are set by indexing a point in the buffer and equating it to a four member list. `buf[0] = [rr,ir, ri,ii]`

Experiments that co-evolve three indirect dimensions contain 8 quadrature components and the data order is rrr, irr, rir, iir, rri, iri, rii, iii, using the same nomenclature as above.

A buffer object has five main attributes, as show in Table 1: `td`, the total number of data points; `nquad`, the number of quadrature components of each data point; `np`, the number of complex, or higher order quadrature, data points; `real`, a list containing all of the real components of a complex buffer and `imag`, a list of the imaginary components of the buffer. As before, the attributes are accessed from the buffer object using the dot operator. For example, `bufobj.np` returns the number of complex points or `bufobj.real` returns a Python list of all of the real quadrature components of complex data.

Table 1 Al NMR Object Attributes

Data attributes	Buffer attributes	Matrix attributes
`dim`–number of dimension	`td`–total data points	`dim`–number of dimensions
`td`–total data points	`np`–number of complex points	`td`–total data points
`o1`–carrier offset (Hz)	`nquad`–num. of quadrature	`bf`–base frequency (Mhz)
`bf1`–base frequency (Mhz)	`real`–real data	`sw`–sw (ppm)
`nuc`–nucleus	`imag`–imaginary data	`ref`–reference shift (ppm)
`sfo1`–carrier frequency (Mhz)		`label`–dimension label
`sw_h`–sweep width (hz)		
`sw`–sweep width (ppm)		

Standard mathematical operations are available for buffer objects, including addition, subtraction, multiplication and division. The operations are performed on an element basis iterating through each element of the buffer.

Table 2 lists the buffer object member functions that are used for data processing. Again, member functions are accessed using the dot operator. A complete list of the functions including the argument keywords and descriptions are included in Online Resource 1. The usage of these functions is consistent with traditional methodology (Hoch and Stern 1996). In addition to the standard data processing functions Al NMR also expands the capability of these functions for processing radial and random sampled data. Al NMR also includes a set of functions unique to radial and random sampled data. The processing functions for co-evolved data are listed in Table 2, where the a subscript indicates that a function processes both traditional and sparse sampled data. A function for processing radial sampled data is available for each step of the processing protocol outlined in the introduction. To facilitate understanding of these new functions an example of typical application is presented here. As described below, a vector is read from a matrix object into a data buffer to initiate processing. Typically the data is stored with interleaved quadrature components, the data is converted to a four quadrature component buffer using the `quad` command. For subsequent operations the order of a four quadrature component buffer is rr, ir, ri, ii. If necessary a first point correction is applied using the `firstpt` function to correct for a baseline offset resulting from an unequal integral area of the first data point. Each of the quadrature components of the first point is scaled by the supplied correction. Likewise if a receiver increment correction is necessary for data collected with States-TPPI the correction is applied using `StatesTPPI_correct` function. Various apodization functions are available including the shifted sinebell (`ss`), Kaiser (`kw`), exponential (`em`) and Gaussian (`gm`). In the case of radial sampling the sampling angle and weighting for each dimension are supplied as arguments to the functions. For randomly sampled data a list of incremented times is supplied. In addition to apodization functions, Al NMR also incorporates a set of data weighting functions for radial and random sampled data. Appropriately weighting the data is necessary to increase performance of the two-dimensional Fourier transform and assure linearity (Pannetier et al. 2007). In this regard, the qhull program is utilized to calculate the Voronoi cells (Barber et al. 1996). For sparsely sampled data processed with the direct two-dimensional Fourier transform zero-filling is not necessary and the final step of processing is Fourier transformation. The direct two-dimensional Fourier transform explicitly calculates the frequency domain points. The number and frequency of the supplied frequency points dictates the digital resolution of the data points.

There are three versions of the direct two-dimensional Fourier transform available in Al NMR. All three are applicable to radially sampled data and one is applicable to randomly sampled data. Radial sampled data is underdetermined and as a result ridge artifacts extend from the authentic peak chemical shifts. We have shown previously that it is possible to use a combination of matching and non-matching component Fourier transforms to isolate the ridge components (Gledhill and Wand 2007). This provides an important advantage in subsequent processing. Two versions of the 2D-FT isolate the ridge components, `2dftp` and `2dftm`, isolate the positive and negative ridge components respectively. The third version, `2dft`, generates a spectrum with both ridge components. This function is also applicable to randomly sampled data. Additionally, as discussed below, phase correction can also be applied. A two-dimensional memory matrix object is returned from the 2D-FT and is used as a temporary intermediate that can be subsequently stored into the final matrix or plotted in the interface. The returned matrix is n x m points where n is the number of frequency values supplied in the ω_1 list and m is the number of values in the ω_2 list.

Matrix object

The matrix object provides the user with an interface for creating and storing nD matrix files. A versatile indexing

Table 2 Member functions of Al NMR data processing objects

Data object functions	
`open`	Opens NMR data file
`read`	Read a FID
Data processing functions	
`+,-,*,/`	Standard mathematical operations
`abs`[ab]	Absolute value
`complex`[b]	Convert to complex
`conjugate`[b]	Complex conjugate
`conv`[b]	Time-domain convolution[c]
`em`[ab]	Exponential window
`exchange`[ab]	Exchange real and imaginary component
`fft`[b]	Fast fourier transform
`firstpt`[ab]	First point correction[d]
`gm`[ab]	Gaussian window
`hilbert`[b]	Hilbert Transform
`ifft`[b]	Inverse fast fourier transform
`kw`[ab]	Kaiser window
`leftshift`[ab]	Left shift data points
`linebaseline`	Linear baseline correction
`lp`[ab]	Linear prediction[e]
`phase`[ab]	Phase correct[f]
`polybaseline`	Polynomial baseline correction
`polysub`	Polynomial solvent filter[g]
`power`[b]	Power spectrum
`reduce`	Reduce complex to real
`resize`	Resize data buffer
`reverse`[ab]	Reverse order of data points
`rightshift`[ab]	Right shift
`roll`[ab]	Circularly shift data points
`sb`[ab]	Sinebell window
`ss`[ab]	Sinebell squared window
`states_correct`[ab]	Adjust sign of data points
`zerofill`[b]	Zerofill data
Buffer object functions–coevolved data functions	
`avgLP`	Average Linear prediction
`FT2D`	Two-dimensional fourier transform [h]
`FT2Dp`	Radial sampled two-dimenisonal sum ridge FT[f]
`FT2Dm`	Radial sampled two-dimenisonal difference ridge FT[f]
`quad`	Set number of quadrature components
`voronoi`	Voronoi tessellation weighting[i]
Matrix object functions	
`reference`	Reference matrix chemical shifts
`vectdim`	Set the vector mode dimension
`ft_thread`	Fast Fourier transform using multiple cores
`get_pt`	Read point
`get_vector`	Read vector
`set_pt`	Write point
`set_vector`	Write vector
`write2D`	Write two-dimensional memory matrix
`write3D`	Write three-dimensional memory matrix
Matrix artifact removal functions	
`add`	Add two spectra
`LV`	Lower value comparison[j]
`HBLV`	Hybrid back-projection lower value[k]
`clean`	Spectrum clean artifact subtraction[l]
Angle selection	
`angle_select`	Angle selection routine

[a] indicates the function is amenable to both 1D data and coevolved data

[b] indicates the function is available in vector mode

[c] Marion et al. (1989)

[d] Otting et al. (1986)

[e] Barkhuijsen et al. (1985); Gesmar and Led (1988); Zhu and Bax (1992)

[f] Gledhill and Wand (2007)

[g] Callaghan et al. (1984)

[h] Coggins and Zhou (2006); Kazimierczuk et al. (2006a, b); Marion (2006)

[i] Pannetier et al. (2007)

[j] (Kupce and Freeman 2004)

[k] Venters et al. (2005)

[l] Coggins and Zhou (2008); Kazimierczuk et al. (2007)

procedure is available to read and write data to the file. The matrix object also provides a vector mode to optimize data processing speed. The function prototype for creating a matrix object is:

$$\texttt{matobj} = \texttt{matrix}(\text{``path/filename.al''}, (\texttt{d1}, \texttt{d2}, \ldots))$$

The first argument is the new matrix file name including the path. This variable is passed to the function as a Python string. The second argument is a tuple listing the size of each dimension. The length of the tuple determines the dimension of the matrix and there are no limits on the number of dimensions that can be used. The same command is used to open an existing matrix with the exception that matrix dimensions are not supplied. Table 1 lists the matrix objects attributes. Again, the attributes are available using the dot operator. The attributes `dim` and `td` are automatically set when a matrix is created. The remaining attributes are set using the reference function.

Data is accessed, for reading and writing, using a similar indexing procedure as used for buffers. Brackets are used to define the indexing and a value is supplied for

each dimension of the matrix. For example, `matobj[d1,d2,d3]` returns the matrix value at point (d1, d2, d3). A vector within the matrix is returned from the indexing operation if a colon is substituted for one of the dimensions. For example:

$$\texttt{bufobj} = \texttt{matobj}[\texttt{d1}, :, \texttt{d3}]$$

returns, in this case, a d2 vector as a buffer object that is perpendicular to points d1 and d3 of the first and third dimensions, respectively. The same indexing syntax is used for writing values into a matrix. Again, values are passed for each dimension of the matrix and the indexed point is equated to a floating point value that will replace the current value. Vectors and planes of data are set using colons for one or two dimensions, respectively.

To optimize the speed of data processing Al NMR provides a vector mode that is able to automatically process every vector in a given dimension without explicitly defining loops over the matrix. The vector mode function state is initialized by defining the dimension of the matrix to process and the number of points in the dimension to use. This is done using the following function:

$$\texttt{matobj.vectordim}(\texttt{dim}, \texttt{np})$$

Here `dim` is the dimension that will be processed and `np` is the number of points to include from the vector. Data points in a matrix are stored with only a single quadrature component and the real and imaginary components are interleaved in sequential positions. Therefore the `np` value is typically two times the number of complex points. Table 2 lists all of the available functions in vector mode, where a superscript b indicates the processing functions available to vector mode. A separate section lists the functions that are only available to matrix objects, of particular interest is the Fourier transform function. This function has been optimized for multi-core processing and simultaneously processes multiple vectors. Additionally, phase correction terms can be supplied to this function to avoid the Hilbert transform step during a subsequent phase correction.

Frequency domain artifact removal functions are available for sparsely sampled data processed with the direct two-dimensional Fourier transform. The artifact removal functions are listed in Table 2. These functions include both projection reconstruction style comparison functions and the more recently presented clean methods (Coggins and Zhou 2008; Kazimierczuk et al. 2007). The projection reconstruction like functions include back-projection (Coggins et al. 2005; Kupce and Freeman 2004), lower value (Kupce and Freeman 2003) and the hybrid back-projection lower value method (Venters et al. 2005). The clean method generates a model of the artifacts, using a supplied peak list, and then subtracts the model from the spectrum generating a new, artifact reduced matrix.

The Al NMR user interface

Al NMR is available with an intuitive Windows graphical interface for processing and visualizing data and spectra, respectively. A picture of the interface is shown in Fig. 3. The interface contains four frame types: a file selection frame, visualization frames, a parameter frame and a command line frame. The first three frames are primarily responsible for viewing and analyzing spectra while the fourth provides means to interactively process and plot data. Multiple visualization frames can be opened simultaneously.

User created custom processing scripts provide a core capability of the Al NMR processing package. User scripting is based on the Python programming language. Data processing scripts are plain text files and employ the standard Python syntax. Example processing scripts to process a Cartesian sampled 4D NOESY experiment and a (3,2) radially sampled experiment are provide in the Online Resources. Processing script can be run from the command line or through the interface.

Cartesian-type data of any dimension can be processed in the usual way from the command line either using scripts or by manually entering commands sequentially. In what follows, we focus on processing capabilities of Al NMR that are peculiar to the demands of sparsely sampled data with a particular emphasis on radial sampled data.

Phase correction

Using Al NMR it is possible to interactively phase correct classical Cartesian data and radially or randomly sampled data. Interactive phase correction is essential to a complete processing program because it is not always possible to collect a spectrum with pure absorptive line shape. Standard phase correction is not applicable to radially sampled data (Gledhill and Wand 2007). For radially sampled data in which two or more of the indirect dimensions are co-evolved all of the co-evolved dimensions need to be phase corrected simultaneously. Two methods are available to phase correct radial data (Gledhill and Wand 2007). One applies the correction in the time domain while the other applies the correction in the frequency domain. Choosing one method over the other depends on the nature of the data. Phase corrections in the time domain require that the 0 and 90 sampling angles be available. Phase correcting in the frequency domain does not have this requirement but is computationally more expensive.

Phase corrections in the time domain are applied during the direct multidimensional Fourier transform (Gledhill and Wand 2007). This is accomplished by matching the multidimensional Fourier transform to the data. For

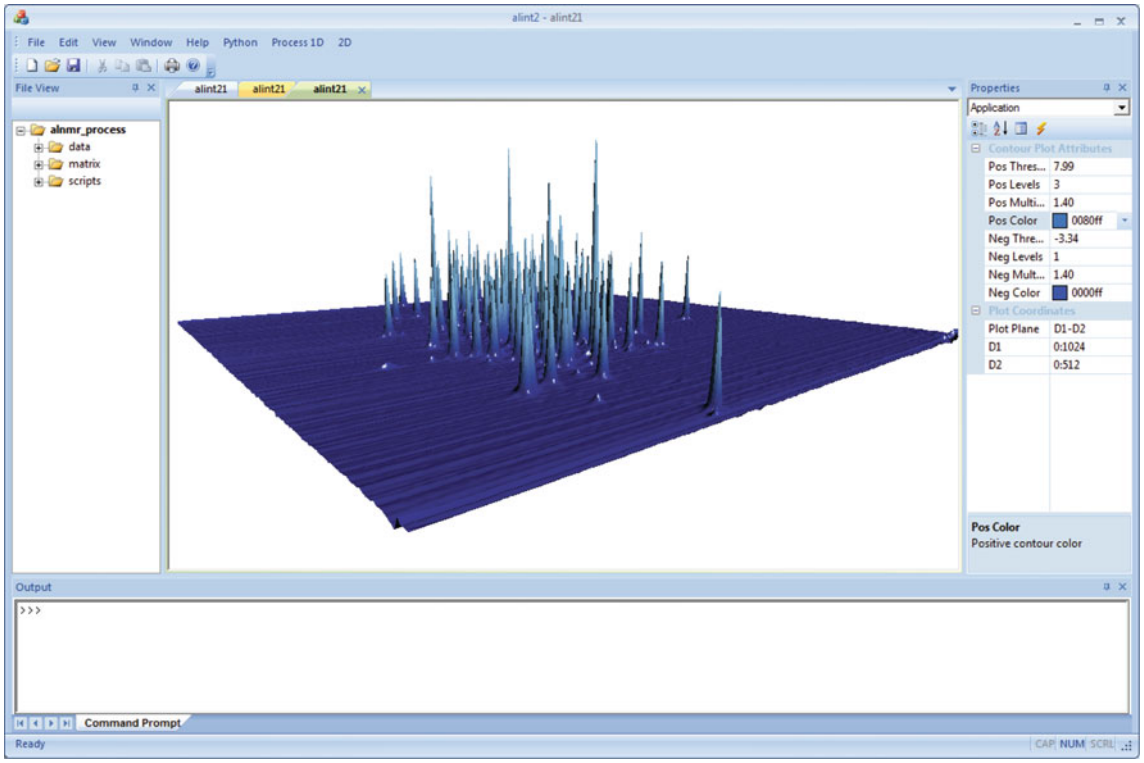

Fig. 3 The Al NMR Windows interface is shown here displaying a stack plot of a ^{15}N HSQC of Ubiquitin. The interface uses a multiple frame layout to make navigation intuitive. The main visualization frame is shown in the center. Multiple visualization frames can be opened simultaneously in new tabs. The file selection frame is shown on the *left*. Expanding the file tree and selecting files will open matrix into the visualization frame or execute scripts in the included interpreter. All of the spectrum display properties are shown in the parameter frame on the *right*. The python command line is show at the *bottom*. The user can interactively process and plot spectra using the command line

example the cosine–cosine modulated quadrature component is Fourier transformed using an explicit Fourier transform with the following coefficients:

$$\cos\left(2\pi\omega_1 \cos\left(t_1 + \phi_1^1\right) + \phi_1^0\right) \cos\left(2\pi\omega_1 \sin\left(t_2 + \phi_2^1\right) + \phi_2^0\right) \quad (1)$$

where ϕ represent the zero and first order phase corrections, as indicated by the superscript term, for the two time domains sampled, defined by the subscript terms. The appropriate phase corrections for each time domain are determined using 0 and 90° sampling angles data sets. The two sampling angle data sets are equivalent to the Cartesian faces of an experiment and are amenable to traditional phase correction methodology. Once the phase corrections are independently determined for each dimension they are used simultaneously as arguments to the two-dimensional Fourier transform.

If the 0 and 90° sampling angle data sets are not available to determine the component phase corrections, the spectrum can be interactively phase corrected in the frequency domain. In this case it is necessary to isolate and phase the ridge components independently because the two ridges that extend from the peak chemical shift have opposite phase correction values (Gledhill and Wand 2007). The ridge components are isolated using a set of matching and non-matching two-dimensional Fourier transforms. In the context of a one dimensional spectrum a matching Fourier transform is defined by applying a real cosine Fourier transform to transform cosine modulated data, while a non-matching Fourier transform is defined as by applying a real sine Fourier transform to transform cosine modulated data. Expanding to two dimensions the Fourier transform can be matching and non-matching to one or both of the time domains sampled. Using this set of Fourier transforms, as defined in previously (Gledhill and Wand 2007), four component spectra are generated and include an absorptive and dispersive spectrum for each of the ridge components. The ridge components are then phased independently in the usual way using linear combinations of the absorptive and dispersive components. The phase corrected ridge component spectra can then be summed to generate a final phased spectrum.

This procedure is accomplished in Al NMR by selecting an indirect plane of the radial sampled angle matrix to use

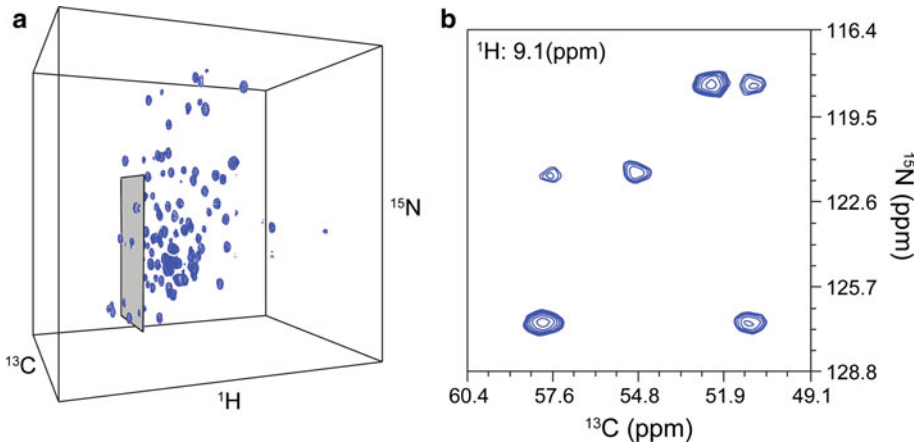

Fig. 4 A visualization of sub-matrix processing is shown here for a radial sampled HNCA processed with the lower value comparison of 17 angles ranging from 5 to 85°. The discrete nature of the direct multi-dimensional Fourier transform allows the user to process any region of the spectrum of interest, including the entire matrix shown in **a** or a selected sub-region as shown in **b**. The selected region shown in **b** corresponds to the *grey slice* highlighted in **a**

for phase correction. The corresponding data vector for the selected plane is then passed to the radial phase correction routine. The routine uses the data vector to generate the four absorptive and dispersive component spectra that are the two pairs of spectra opened in an interactive window to determine the phase corrections. The resulting phase corrections can then be applied to a processing script to phase correct all of the other angle spectra.

Real time sub-matrix processing

One of the significant advantages to sparse sampling is its ability to collect data at significantly higher resolution than the equivalent Cartesian sampled experiments (Kupce and Freeman 2004). As a result the processed matrix size and the necessary processing time can become overwhelming, especially since a large fraction of the spectrum is noise. The discrete nature of the direct two-dimensional Fourier transform can be exploited to process a sub-region of the frequency space. This increases processing speed and eliminates much of the storage requirements to the point that sub-matrix processing can be done in real time and in memory. An example of sub-matrix processing is shown in Fig. 4. Panel a shows the full matrix with the highlighted region corresponding to the region of interest for processing. Panel b shows the sub-spectrum that is processed selectively. The annotated processing script for this approach is provided in the Online Resources.

Angle selection

In order to efficiently collect radial sampled data Al NMR includes an angle selection routine that utilizes recently described algorithms to obtain optimal sets of sampling angles (Gledhill and Wand 2008). There are two general cases in which angle selection is applied; the peak resonance frequencies are known or the frequencies are unknown and need to be resolved. In the former case, a spectrum could be collected in order to measure intensity variations or slight changes in chemical shift in the context of hydrogen exchange or ligand binding for example. Two algorithms are available for this scenario, one to determine a minimum set of angles needed to resolve intensity while disregarding artifact intensity i.e. leaving *resolved* artifact intensity in the final spectrum. The other is comprehensive and selects a minimum set of angles to remove all of the artifacts from the spectrum. In the latter situation where peak resonance frequencies are unknown, Al NMR executes an algorithm to iteratively suggest subsequent sampling angles to resolve all artifacts from a spectrum while retaining the authentic peaks.

To determine if a peak is resolved or if an artifact is removed both algorithms generate a model of the spectrum. Using the chemical shifts and sampling angle the information is modeled as a set of linear equation that describe the peaks as a point and the artifact ridges as a line defined by the following equation.

$$p = p_1 + n(0, \cos(\pm 90 - \alpha), \quad \sin(90 - \alpha)) \tag{2}$$

Here p represents the ridge vector, p_1 is the original peak chemical shift, n is an arbitrary value and α is the sampling angle.

With the above definition it is possible to apply linear algebra and scale for the line width of peaks to determine if a peak is uniquely resolved from all artifacts. This is done by calculating a peak to ridge distance. If the minimum distance from a peak to the artifact ridge extending from every other peak is greater than a user defined cutoff the peak is considered resolved because no artifact intensity obscures the intensity of the peak. A set of angles that resolve all of the peak intensities is determined by iteration of this algorithm over potential sampling angles and sorting for the most productive combination. Further details may

be found elsewhere (Gledhill and Wand 2007). In practice angle selection is accomplished using the `angle_selection` function.

Statistical sensitivity enhancement

Al NMR contains all of the necessary features necessary to implement SEnD optimization (Gledhill and Wand 2010). SEnD is the strategy by which radially sampled data collection is optimized to maximize the signal-to-noise of the final spectrum. The SEnD procedure exploits the statistical properties of radially sampled data sets, namely multiple independent data sets are collected with redundant information. SEnD optimization results from the comparison of this redundant information and relies on each sampling being of sufficient signal-to-noise to guarantee the survival of an authentic peak during lower-value comparison. Generally, a signal-to-noise of 6 is necessary to retain all peaks in a spectrum (Gledhill and Wand 2010). The acquisition parameters necessary to achieve this threshold can be determined from lower dimensional projections of the experiment.

Computational efficiency

Al NMR processes Cartesian sampled data with efficiency comparable to traditional processing programs. For example, approximately 0.5 and 10 s are consumed while processing a 2D gradient selected HSQC data set to a final matrix size of 1024 × 512 and a 3D HNCO data set to a final matrix size for 1024 × 256 × 256, respectively. Processing the same data with NMRpipe requires 5 and 13 s, respectively. Processing a single sum or difference spectrum with the two-dimensional Fourier transform requires approximately 40 s. Complete processing of a radial sampled HNCO comprised of the sum and difference spectra for 17 sampling angles, an additive back-projection equivalent spectrum and a lower value comparison spectrum were generated in approximately 30 CPU min using a single processor. This time can be reduced to approximately 6 min by calculating all of the spectra in parallel using a standard python multithreaded approach in the processing script. All processing was preformed using an Intel I7–920 processor.

Summary

A new processing program based on the Python scripting language is presented here. The clear syntax of Python is easy to learn and many resources are available to aid in future development. A complete set of NMR data processing functions for both Cartesian and radially sampled data are available in Al NMR. The Python framework allows flexibility in processing and implementation of custom functions. Al NMR is particularly suited to the processing of sparsely sampled data and enables the essential processing tasks that are not possible with traditional approaches. These include sampling angle set selection, phase correction, sub-matrix processing, ridge artifact suppression, sensitivity enhancement through statistical analyses, and ridge peak integration. The executable version of Al NMR is available for non-commercial use and may be downloaded from www.wandlab.org.

Acknowledgments We thank Vignesh Kasinath and Kathleen Valentine for helpful discussion. This work was supported by NIH grants DK 39806 and GM 081520, by NSF grants MCB 0842814 and DMR 05-20020 and by a grant from the Mathers Foundation.

References

Anderson E (1999) LAPACK Users' Guide. Software, environments, tools, 3rd edn. Society for Industrial and Applied Mathematics, Philadelphia

Barber CB, Dobkin DP, Huhdanpaa H (1996) The quickhull algorithm for convex hulls. ACM T Math Software 22:469–483

Barkhuijsen H, Debeer R, Bovee WMMJ, Creyghton JHN, Vanormondt D (1985) Application of linear prediction and singular value decomposition (LPSVD) to determine NMR frequencies and intensities from the FID. Magn Reson Med 2:86–89

Bodenhausen G, Ernst R (1982) Direct determination of rate constants of slow dynamic processes by two-dimensional "Accordion" spectroscopy in nuclear magnetic resonance. J Am Chem Soc 104:1304–1309

Brutscher B, Morelle N, Cordier F, Marion D (1995) Determination of an initial set of NOE-derived distance constraints for the structure determination of 15 N/13C-labeled proteins. J Magn Reson B 109:238–242

Callaghan PT, Mackay AL, Pauls KP, Soderman O, Bloom M (1984) The high fidelity extraction of weak broad lines from NMR-spectra containing large solvent peaks. J Magn Reson 56:101–109

Coggins BE, Zhou P (2006) Polar Fourier transforms of radially sampled NMR data. J Magn Reson 182:84–95

Coggins BE, Zhou P (2007) Sampling of the NMR time domain along concentric rings. J Magn Reson 184:207–221

Coggins BE, Zhou P (2008) High resolution 4-D spectroscopy with sparse concentric shell sampling and FFT-CLEAN. J Biomol NMR 42:225–239

Coggins BE, Venters RA, Zhou P (2005) Filtered backprojection for the reconstruction of a high-resolution (4, 2) D CH3-NHNOESY spectrum on a 29 kDa protein. J Am Chem Soc 127:11562–11563

Coggins BE, Venters RA, Zhou P (2010) Radial sampling for fast NMR: concepts and practices over three decades. Prog Nucl Magn Reson Spectrosc 57:381–419

Delaglio F, Grzesiek S, Vuister GW, Zhu G, Pfeifer J, Bax A (1995) NMRpipe—A multidimensional spectral processing system based on unix pipes. J Biomol NMR 6:277–293

Eghbalnia HR, Bahrami A, Tonelli M, Hallenga K, Markley JL (2005) High-resolution iterative frequency identification for

NMR as a general strategy for multidimensional data collection. J Am Chem Soc 127:12528–12536
Freeman R, Kupce E (2003) New methods for fast multidimensional NMR. J Biomol NMR 27:101–113
Frigo M, Johnson SG (2005) The design and implementation of FFTW3. Proc IEEE 93:216–231
Gesmar H, Led JJ (1988) Spectral estimation of complex time-domain NMR signals by linear prediction. J Magn Reson 76:183–192
Gledhill JM, Wand AJ (2007) Phasing arbitrarily sampled multidimensional NMR data. J Magn Reson 187:363–370
Gledhill JM, Wand AJ (2008) Optimized angle selection for radial sampled NMR experiments. J Magn Reson 195:169–178
Gledhill JM, Wand AJ (2010) SEnD NMR: sensitivity enhanced n-dimensional NMR. J Magn Reson 202:250–258
Gledhill JM, Walters BT, Wand AJ (2009) AMORE-HX: a multidimensional optimization of radial enhanced NMR-sampled hydrogen exchange. J Biomol NMR 45:233–239
Hiller S, Fiorito F, Wuthrich K, Wider G (2005) Automated projection spectroscopy (APSY). Proc Natl Acad Sci USA 102:10876–10881
Hoch JC, Stern AS (1996) NMR data processing. Wiley-Liss, New York
Hoch JC, Maciejewski MW, Filipovic B (2008) Randomization improves sparse sampling in multidimensional NMR. J Magn Reson 193:317–320
Jaravine V, Ibraghimov I, Orekhov VY (2006) Removal of a time barrier for high-resolution multidimensional NMR spectroscopy. Nat Method 3:605–607
Kazimierczuk K, Kozminski W, Zhukov I (2006a) Two-dimensional Fourier transform of arbitrarily sampled NMR data sets. J Magn Reson 179:323–328
Kazimierczuk K, Zawadzka A, Kozminski W, Zhukov I (2006b) Random sampling of evolution time space and Fourier transform processing. J Biomol NMR 36:157–168
Kazimierczuk K, Zawadzka A, Kozminski W, Zhukov I (2007) Lineshapes and artifacts in multidimensional Fourier transform of arbitrary sampled NMR data sets. J Magn Reson 188:344–356
Kazimierczuk K, Zawadzka A, Kozminski W (2008) Optimization of random time domain sampling in multidimensional NMR. J Magn Reson 192:123–130
Kazimierczuk K, Stanek J, Zawadzka-Kazimierczuk A, Kozminski W (2010a) Random sampling in multidimensional NMR spectroscopy. Prog Nucl Magn Reson Spectrosc 57:420–434
Kazimierczuk K, Zawadzka-Kazimierczuk A, Kozminski W (2010b) Non-uniform frequency domain for optimal exploitation of non-uniform sampling. J Magn Reson 205:286–292
Kim S, Szyperski T (2003) GFT NMR, a new approach to rapidly obtain precise high-dimensional NMR spectral information. J Am Chem Soc 125:1385–1393
Kupce E, Freeman R (2003) Reconstruction of the three-dimensional NMR spectrum of a protein from a set of plane projections. J Biomol NMR 27:383–387
Kupce E, Freeman R (2004) Projection-reconstruction technique for speeding up multidimensional NMR spectroscopy. J Am Chem Soc 126:6429–6440
Marion D (2006) Processing of ND NMR spectra sampled in polar coordinates: a simple Fourier transform instead of a reconstruction. J Biomol NMR 36:45–54
Marion D, Ikura M, Bax A (1989) Improved solvent suppression in one-dimensional and two-dimensional NMR-spectra by convolution of time-domain data. J Magn Reson 84:425–430
Otting G, Widmer H, Wagner G, Wuthrich K (1986) Origin of T1 and T2 ridges in 2d NMR-spectra and procedures for suppression. J Magn Reson 66:187–193
Pannetier N, Houben K, Blanchard L, Marion D (2007) Optimized 3D-NMR sampling for resonance assignment of partially unfolded proteins. J Magn Reson 186:142–149
Stern AS, Li KB, Hoch JC (2002) Modern spectrum analysis in multidimensional NMR spectroscopy: comparison of linear-prediction extrapolation and maximum-entropy reconstruction. J Am Chem Soc 124:1982–1993
Szyperski T, Wider G, Bushweller JH, Wuthrich K (1993) Reduced dimensionality in triple-resonance NMR experiments. J Am Chem Soc 115:9307–9308
Venters RA, Coggins BE, Kojetin D, Cavanagh J, Zhou P (2005) (4, 2)D projection-reconstruction experiments for protein backbone assignment: application to human carbonic anhydrase II and calbindin D-28 K. J Am Chem Soc 127:8785–8795
Zhu G, Bax A (1992) Improved linear prediction of damped NMR signals using modified forward backward linear prediction. J Magn Reson 100:202–207