The ssNake reference manual

Wouter Franssen & Bas van Meerten

Version 0.2

29th October 2015

Contents

1 Running ssNake

1.1 Python versions

ssNake had been programmed to run on both the python 2.x and 3.x branch. We developed ssNake using python 2.7.9 and 3.4.3, but somewhat older version might also be sufficient.

1.2 Numpy versions

We have run into issue while using older versions of numpy. It seems that version 1.7.0 is the oldest that we support (some new commands have been introduced in this version, that ssNake makes use of). Development took place on version 1.8.2.

2 File

Using the ssNake load tools is quite easy. Go to File → Load. Navigating and double clicking on the desired file then loads the data. Many formats supported by ssNake (like Bruker and Varian) have their data in a folder, in which several files with a fixed name are present. For these formats, loading any of the files in this directory is accepted (ssNake searches for the expected files in the selected directory). Loading large data files might take some time, depending on your computer hardware.

When loading the data, the user is prompted to give in a name for the data by which it is identified in the workspace selector (see the Worspaces section for more on this).

2.1 Formats

ssNake is able to extract NMR data from a number of formats from several vendors. The following table summarizes the support.

Name	Specification
Varian/Agilent	VnmrJ 2 and newer
Bruker	Topspin and XWinNMR (fid & ser)
Chemagnetics	???
Magritek	Both 1D and 2D data
Simpson	1D and 2D, -ascii and -binary
JSON	ssNake JSON file (ascii)
Matlab	ssNake .mat file (binary)

Generally, ssNake loads the data raw data and searches for some variables (e.g. spectral width and spectrometer frequency). The above data formats do not support data with more then two dimensions in a nice way. Getting the higher dimensions is therefore not straightforward. ssNake treats this data as 2D, after which you can split the data yourselves using Matrix → Split.

Varian/Agilent

Loading Varian/Agilent files requires a *procpar* file and a *fid* file. From the *procpar*, the spectral frequency and the spectral width in one or two dimensions is extracted. The *fid* file is then checked for the version (VnmrJ 2 or 3) and the data is extracted from this file.

Bruker

The Bruker load first checks for the *acqus* and *acqu2s* files and extracts the number of points in both dimensions, the spectral widths and the spectral frequency. Also, the bitorder is checked, which describes the way the data is saved in the *fid* or *ser* file. Using this, the data is extracted.

Chemagnetics

Loading Chemagnetics data requires a *acq* (and *acq_2* for 2D) and *data* file. From the former, the number of points in both dimensions is extracted, the spectral

widths and the spectral frequency. All the data points are then extracted form the *data* file, and reshaped to a 2D array when necessary.

Magritek

Magritek requires two or three files: *acqu.par* for the spactral variables, and a file that ends with *.1d*. For 2D data, a file that ends with *.2d* is also needed. As with the other formats, the number of data points in the two dimensions is extracted from the parameter file (along with the spectral data) and the binary data in the *.1d* or *.2d* is unpacked.

Simpson

Simpson data files are plain text files, *.fid* or *.spe* as file extension. From the file, the number of points in both dimensions is extracted, and the spectral widths. The spectral frequency is *not* included in the Simpson format, and is put at 0. Simpson files cannot contain a mixture of time and frequency data (both dimensions must be the same type). Simpson binary data is also supported, but is very slow to load.

JSON

Loads a JSON file that is saved with ssNake. File must have the *.json* or *.JSON* extension.

Matlab

Loads a Matlab file that is saved with ssNake. File must have the *.mat* or *.MAT* extension.

Unsupported file formats

Currently, we support all the data formats that we have access to. While we have some code lying around for loading JEOL data, for example, we cannot include this if we cannot test it. If you want your favourite data format to be support by ssNake, please send us a request along with some sample data (1D and 2D).

2.2 Load

When using ssNake's load function, ssNake analyses the selected file name and the names of other files in the directory to figure out the format you want to load. ssNake only needs to check for files that are actually needed for loading the data, so removing useless files from the format does not pose a problem. Below, the checks used by ssNake in this function can be found for each format.

Name	Folder contains files	File extension
Varian/Agilent	procpar and fid	
Bruker	acqus (and acqu2s for 2D) and fid or ser	
Chemagnetics	acq (and acq_2 for 2D) and data	
Magritek	acqu.par and *.1d or *.2d	
Simpson		.fid or .spe
JSON		.json or .JSON
Matlab		.mat or .MAT

2.3 Save data

Naturally, ssNake can also be used to save data. When saving data, the current ND data is saved, along with the spectral widths, frequencies and if the axis is in time or frequency units. Ppm references and the current view are *not* saved. Saving to Simpson format also has some restrictions as pointed out below.

JSON

JSON (JavaScript Object Notation) is a ascii format (i.e. regular text) used to save data structures. Within ssNake this can be used to save the current data in a clear, human readable format. As JSON data is in ascii, it is not efficient in file size and in speed. If these are necessary then consider saving the data as a matlab binary.

MATLAB

The MATLAB binary format is a open source format in which many different types of data can be saved. Also, it is the native format of MATLAB, which can be used for more special data manipulation, if necessary. As the format is binary,

it is efficient in both speed and file size. The content of the file is the same as the JSON file.

Simpson

Simpson is a general NMR simulation program with its own data format. The format only supports 1D and 2D data, and only if the data in both dimensions is of the same type (i.e. both spectrum, or both time). Apart from that, Simpson is a ascii format, and therefore has the same speed restrictions as the JSON format. The parameters that are saved within the Simpson data are restricted (no offset frequency), so when loading it back into ssNake some information is lost. We therefore advise to use the Simpson format only for transferring data to another program.

2.4 Save Figure

The Save Figure option can be used to save any Figure plot within ssNake. The title, axis labels and size of the Figure can be changed. Be aware that choosing a to small height can cause the x-axis label to disappear. After all the titles have been set, pushing 'Save' will save the Figure to a directory of your choice. The Figure can be saved in vector format (.eps, .pdf, .svg), in lossless pixel format (.png), and in lossy pixel format (.jpg). We advice to use vector formats to make high quality pictures.

3 Workspaces

Within ssNake you can have multiple data files opened. Each loaded file gets its own environment in which you can process the data. These are called workspaces. All ssNake workspaces are independent, meaning that the data from workspace 1 cannot be changed while viewing any other workspace. Going back to a workspace from another goes back to the same view as before: nothing is discarded when switching between workspaces. Also: every workspace has its own undo/redo information. Some tools can transfer data from other workspaces to the current workspace. You can find more about these tools later on, in their respective sections.

When loading data, ssNake prompts for a name. This will become the name of the workspace in which the data is loaded. The title of the graph will be

set to this name, and the name will be set in the list of opened workspaces in Workspaces → Active. Note that you cannot have multiple workspaces with the same name.

3.1 Duplicate

Makes a copy of the current workspace to another workspace, of which the name is asked. All the data and the current view is copied to this new workspace. No undo and redo data is transferred. Also, the current workspace is now the new, duplicated one.

3.2 Delete

Deletes the current workspace and all references to it. The view is changed to the next workspace in the list. Be aware that there is no way to undo this!

3.3 Rename

Renames the current workspace. Note that two workspaces cannot have the same name.

3.4 Active

Shows a list of all workspaces and indicates the active one with a dot in front. The current workspaces can be changed by clicking on a not active one.

3.5 Keyboard shortcuts

ssNake features several keyboard shortcuts to work with the workspaces. Below is a list of all the combinations.

Combination	Effect
Ctrl + w	Close current workspace
Ctrl + d	Duplicate workspace
Ctrl + Page Up	Change current workspace to the one above in the list
Ctrl + Page Down	Change current workspace to the one below in the list

When the current workspace is the last in the list, and Ctrl + Page Down is pressed, the current workspace is set to the first in the list.

4 Macros

Macros can be used to save a particular series of action performed on the data, for later fast reuse. Say that you want to load several data files, and want to set the size of them to 4096 points and perform a Fourier transform. Doing this many times can be annoying. When you create a macro to do this, you only need to execute the macro for every data file. Thus saving time and making sure all are precessed in an identical way. Also, macros are shared between workspaces: there is only one pool of macros. How to use macros is explained below.

4.1 Start recording

To create a macros, you must tell ssNake that you want to record a series of action, to be saved in the macro. Executing Macros → Start Recording prompts for a name of the future macro, and enables the recording. From now on, all the actions you perform on the data are recorded and saved in this macro.

4.2 Stop recording

After some actions have been performed (Fourier, zero fill, phasing...) the recording of the macro can be stopped using Macros → Stop Recording.

4.3 Run

Can be used to run the selected macro on the current data.

4.4 Delete

Delete a macro.

4.5 Save

Can be used to save a macro to the disk as a .json file. This is a plain text file, in which you can see all the steps that are executed when running this macro.

4.6 Load

Loads a saved macro in ssNake. Execution of the macro can then be performed using Macros → Run.

4.7 Notes on usage of macros

Note that ssNake only records the actions that are not undone. If you perform an action when recording is one, and you undo it while recording, the macro has no entries. However, performing two actions that cancel each other *are* recorded. For example, issuing Fourier twice has no net effect, but a macro of this includes both commands.

SOME PART ABOUT RUNNING MACROS ON MORE OR LESS DIMENSIONS THEN THEY WERE CREATED FOR.

5 Edit

5.1 Undo

In ssNake, every action that causes a change in the data can be undone. When such an action is performed, the undo list is increased with one entry, in which the information is stored as how to undo the performed action. Sometimes this involves the inverse operation (e.g. for fft/inverse fft), but for some actions the old data must be preserved and put back when an undo is requested. Duplicating data to another workspace does not copy the undo list.

Undo cannot be used the undo the current view to the previous view as this is not an operation that changes the data.

5.2 Redo

When Undo is used, the action that is undone is put in the redo list. Using Undo and then Redo therefore performs no net action. Every time a new action (i.e. something other then redo) is performed the redo list is cleared.

5.3 Reload

This action reloads the data from the original source. All undo and redo information is cleared. Can be useful if you use some external program to change the source, and you want ssNake to update its data. Creating a macro of the actions you perform on this data might be useful in this case.

6 Tools

6.1 Real

Real puts the imaginary values of all data points at zero.

Background

In some NMR experiments, only the real value of the magnetization during a specific period is measured. In this case it is convenient to zero the imaginary part to force further processing to ignore this data.

6.2 Imag

Imag puts the real values of all data points at zero.

6.3 Abs

Abs replaces all complex datapoints by the length of the vector they span in the 2D complex plane. That is:

$$F_{\text{new}} = \sqrt{\text{real}(F_{\text{old}})^2 + \text{imag}(F_{\text{old}})^2}$$

for all points in the complex data.

6.4 Sizing

The sizing tool can be used to edit the number of points in the current dimension.

Background

The number of point taken in an NMR time signal is usually restricted, as increasing the acquisition time leads to more noise (as the signal has decayed) or adds to much strain on the device (if decoupling is used). As the number of points in the spectrum is equal to the number of points in the time signal, this limits the *digital* resolution of the spectrum, making it less easy to extract the desired data. However, if the signal has decayed at the end, the digital resolution in the spectrum can be enhanced by adding zeroes at the end, without effecting the spectrum in terms of signal-to-noise and appearance. This is termed ‘zero filling’ and is often used in NMR.

If the time signal has not fully decayed, the signal abruptly changes to zero. This leads to distortions in the spectrum (i.e. sinc wiggles) as the time signal is

effectively multiplied with a block function. General Fourier theory then states that the spectrum must then be convoluted with the Fourier transform of the block function, which is $\sin(x)/x$, i.e. sinc.

ssNake implementation

Selecting 'Tools → Sizing' creates a sizing window. In here a single number has to be filled in the box. Decimal values will be rounded to the nearest integer. Apart from numbers, the command 'k' (or 'K') is also supported, which stands for '1024'. When using this definition there has to be a number before the 'k'. A valid input would therefore be: '2k'. Which sets the number of points to $2 \cdot 1024 = 2048$. Pressing 'Apply' executes the sizing command on all traces in all dimensions.

If the supplied number is larger than the current size, zeroes are added at the right-hand side of the time signal. If the value is lower, points are removed from the right-hand side of the time signal. If the whole echo type is on, the zeroes are added in the centre of the time signal, or points are taken symmetrically from the centre, if the size is reduced.

When applied to the frequency domain, zero filling makes no real sense. However ssNake does support this. In this case zeroes are added to the left-hand side of the spectrum (due to the mirroring of the spectrum used in NMR), or taken from them. When doing this, the spectra width should be changed to be still correct. This is not done, and it is therefore not recommended to use this option in the frequency domain.

6.5 Swap echo

A tool to swap the time domain data symmetrically around a selected point.

Background

In some NMR experiments, the rise and fall of a spin echo is recorded. One way of processing this data is to remove all the data points before the echo maximum, and subsequently treat the data as a regular FID. In some cases it is more useful to use the full data, rise and fall, of the echo signal. Directly Fourier transforming such data leads to massive first order phase distortion. This can be circumvented by swapping the echo around the centre. In this case, the data is split in two parts: the data from the start of the measurement until the top of the

echo, and the data from the top of the echo until the last point. The latter data is put leftmost, the former right. The new signal then starts with a decay and rises again at the end. Due to the symmetry of the echo, the Fourier transform of this signal will have zero imaginary signal, while the real signal is still the same as if a regular FID is transformed. This means that phasing such a line can be easily done.

ssNake implementation

Echo swapping is done by left clicking on the echo maximum of the time signal. By default, the swapping is set at the centre of the time signal. After left clicking, a preview is shown. Pushing 'Apply' applies the transformation to all traces of this dimension. Also, the tag 'Whole echo' is set in the footer. When this is on, Fourier transforming does not multiply the first point by 0.5 (as is needed for data that decays to zero), and line broadening is applied symmetrically around zero.

6.6 Shift data

Shifts the data to the left or to the right, by removing data points on the left or on the right, and filling with zeroes on the other side.

Background

Data shifting can be used if some data points on the left or right of the time signal are unnecessary or unwanted. When measuring a spin echo, for example, it is not uncommon to start recording the signal some microseconds before the expected echo top, to make sure the full signal is measured. The data can then be left shifted to remove the data points before the echo maximum, and yield a regular spectrum.

ssNake implementation

Starting the shift data tool creates an input window. Within this window, a integer number must be filled in that equals the number of data points of the desired *right* shift. Left shifts can be made by using negative values. Alternatively, the arrows on the right and left side of the input box can be used to shift the data in the direction of the arrow. The effect of the operation on the current 1D

can be seen in the main window. When pushing 'Apply' the shift is performed on all traces of the nD data.

ssNake can also perform the time domain shift from within the frequency domain. When in a spectrum, an inverse Fourier transform is used to go back to the time domain. The data is then shifted with the desired measure, and a forward Fourier transform gives the spectrum of the shifted time signal. The effect of the sift on the current 1D can be seen in the main window. As this calculation is a bit more involved, it can take some time if the data set is large.

6.7 Offset correction

Subtracts the average value of the selected data points from all data points.

Background

It can sometimes happen that NMR data does not decay to zero, but to another value. In this case, a time signal with zero frequency and no decay can be thought seems to be added to the NMR signal. Usually, this is due to issues with the electronics, and does not imply a NMR signal with these properties. For proper analysis of the signal, this offset should be removed, as it leads to a signal at 0 Hz, which can interfere with the spectrum. This can be solved by subtracting the average value of the last points of the time domain data from all the data points.

ssNake implementation

Using the Offset correction tool opens a new window. By default, the last 20% of the time data is selected as containing only noise (of which the average is the offset which should be subtracted). Left clicking two times in the graph sets the limits between which the average value is calculated. Pushing apply subtracts this value from all time data points. If the data has more dimensions then 1, the average value is subtracted from all data points. Take note that the average is only calculated for the current graph. If you want to correct the offset for each graph you can select the 'Single slice' option, and do the correction separately in each desired trace.

6.8 Baseline correction

Fits a polynomial line of a given order through the current spectrum while ignoring the selected parts. This line is then subtracted from the spectrum.

Background

In some NMR spectra, experimental conditions were such that a non-informative baseline is present in the spectrum. This can be due to electronic reasons (e.g. pulse ringing) or due to background signals. This usually means that the first points in the time signal are corrupted. In some cases, removing the first data points is the best option, but this leads to signal loss and a large first order phasing error. In these cases, it can be more convenient to try to subtract the errors in the frequency domain. This is done by fitting a nth degree polynomial through the spectrum, and subtract this line from the spectrum. For this, it is essential that the regions of spectral information (i.e. the NMR peaks) are not included in the fit. Failing to do this will cause the baseline correction to attempt to remove the peaks.

Baseline correction is mostly a visual thing. In the case of ssNake, the degree of the polynomial to be subtracted has to be put in. Usually, putting a very high value leads to strange results. This, however, depends on the spectrum to be corrected.

ssNake implementation

In ssNake, baseline correction is performed by fitting a nth degree polynomial through the spectrum, i.e.:

$$S_{\text{corr}} = \sum_{i=0}^{n} a_i x^i,$$

with a_i the amplitude of the ith degree polynomial. Fitting this to the spectrum is a mathematical operation with a single unique solution. Prior to doing this, specific parts of the spectrum can be selectively ignored in the fitting procedure. This can be done by left clicking in the spectrum at the two limits of the area that is to be ignored. The area is then marked with a red background. This way, multiple areas can be selected. Pressing ‘Fit’ then performs and shows the fir. Apply subtracts the line from the spectrum. Note that by default this single fit is subtracted from *all* the traces. This is useful if the background is identical in

all of them. Alternatively, the box 'Single slice' can be ticked to only apply the correction on the current spectrum.

6.9 Correct Bruker digital filter

Corrects a Bruker fid to start at $t = 0$.

Background

In theory, the recording of an NMR signal starts directly after the excitation point. The signal can then be easily described by a decaying exponential (or another function) and the Fourier transform does not show surprises. NMR data measured on a Bruker device, however, has the peculiar attribute that it starts *before* the physical start of the signal. The origin of this lies in the way Bruker spectrometers use their digital filter, of which de details will not be described here. Direct Fourier transformation of this signal will lead to massive oscillations in the baseline of the resulting spectrum.

To be able to handle Bruker NMR data, this detection error has to be corrected. The preferred way for this seems to apply a strong first order phase correction in the spectrum that (due to Fourier principles) rolls that time domain data in such a way that the first point is now the start of the real signal. Due to this procedure, the points that were at the front of the time signal are now at the end. Apparently this leads to better results than just removing this data.

ssNake implementation

To automate this process, ssNake uses information from a handy text by W. M. Westler and F. Abildgaard. Every Bruker NMR spectrometer has its own characteristics, and therefore needs a different amount of first order phase correction. For older Bruker hardware, the amount of correction is found by extracting two parameters from the Bruker acqus file, and to use a lookup table supplied by Westler & Abildgaard. For newer hardware, this value can be read directly from the acqus file.

Using this tool opens a windows, in which you have to open (again) the current Bruker file. ssNake then finds the desired phase correction and applies it to all traces.