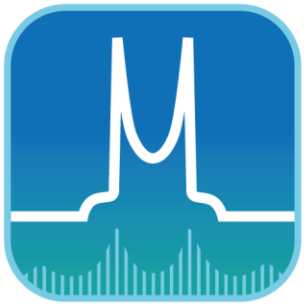


SolidSpin

Solid-state NMR Simulation Plugin

Contents

This document is intended to give the user an overview of JASON solid-state NMR simulation plugin and to allow them to orient themselves within the software. It is assumed that the user has a working knowledge of NMR processing and JASON. JASON is running under the Windows operating systems. macOS instructions are not included but the operation should be similar.

1 Introduction

SolidSpin is a plugin which provides an easy-to-use graphical user interface for the general solid-state NMR simulation program: SIMPSON. SIMPSON is described elsewhere: https://doi.org/10.1006/jmre.2000.2179.

SIMPSON is a command-line utility. The input is based on Tcl files which describe the simulation. A wide range of solid-state (and beyond) NMR problems can be simulated on the quantum-mechanical level.

The described JASON plugin uses modified versions of these input files. Modifications include a section which describes the GUI within the JASON plugin, as well as code to exchange the parameters between JASON and SIMPSON.

Some examples of the input files are included with the plugin. Users can modify, create and use their own input files for specific problems.

Exact general quantum-mechanical simulations may not be the most optimal solution for some specific cases, which may have simplified, often analytically defined, solutions. But this approach provides a high level of flexibility and power for solid-state NMR analysis while maintaining reasonable performance.

Below you will find instructions on setting up the plugin and how to use it for two major tasks:

1. Simulation of 1D solid-state NMR spectra
2. Finding NMR parameters by fitting experimental 1D solid-state NMR spectra with simulated spectra

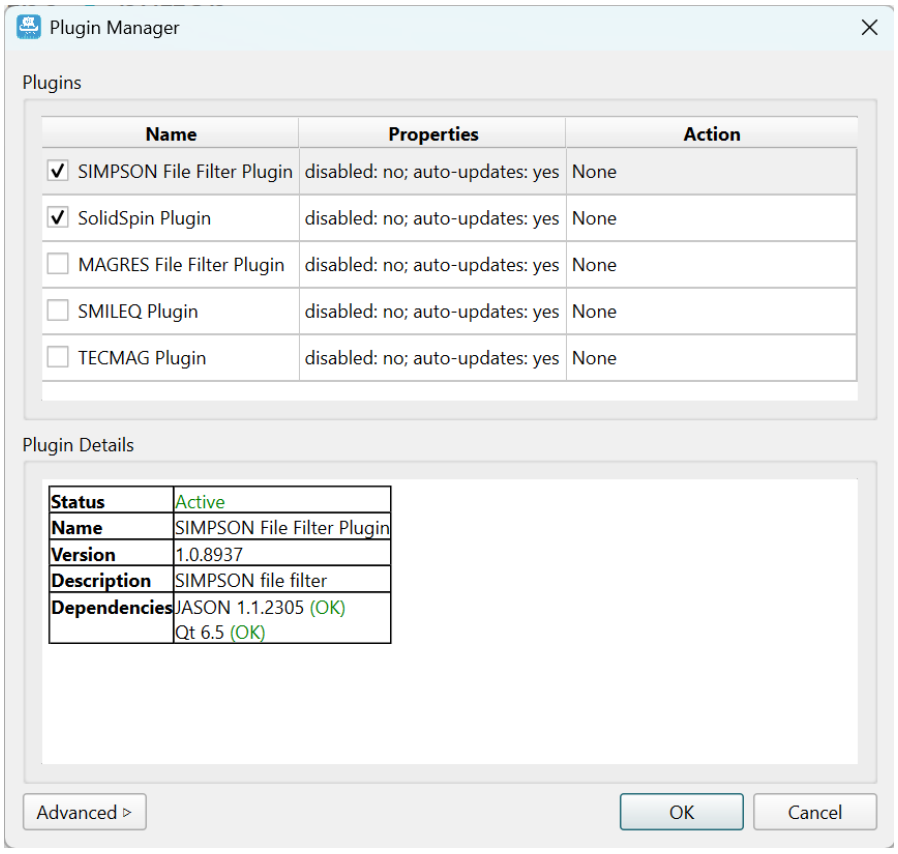
2 Installation and setup

Plugin installation consists of two steps: activating both the SolidSpin NMR simulation and SIMPSON file format plugins in Plugin Manager, and then configuring the plugin for use.

2.1 Activation in Plugin Manager

The Plugin Manager is accessed via the three dots Main Menu button and displays all the available plugins for the opened instance of JASON.

The plugins have to be activated before use. The SolidSpin plugin is supplied together with JASON. You should find it in the list of available plugins in the Plugin Manager. It should be activated by checking the check box next to “SolidSpin Plugin” in the list. Additionally, the “SIMPSON File Filter Plugin” is needed to load files generated by SIMPSON. It has to be activated as well:

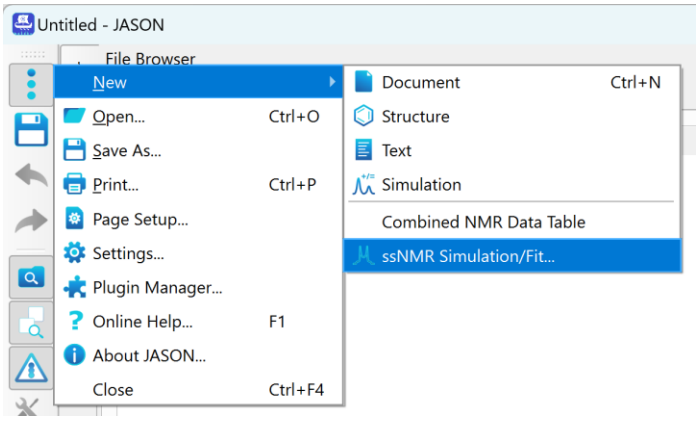


After clicking OK you will be prompted to restart JASON so the newly activated plugins can be loaded.

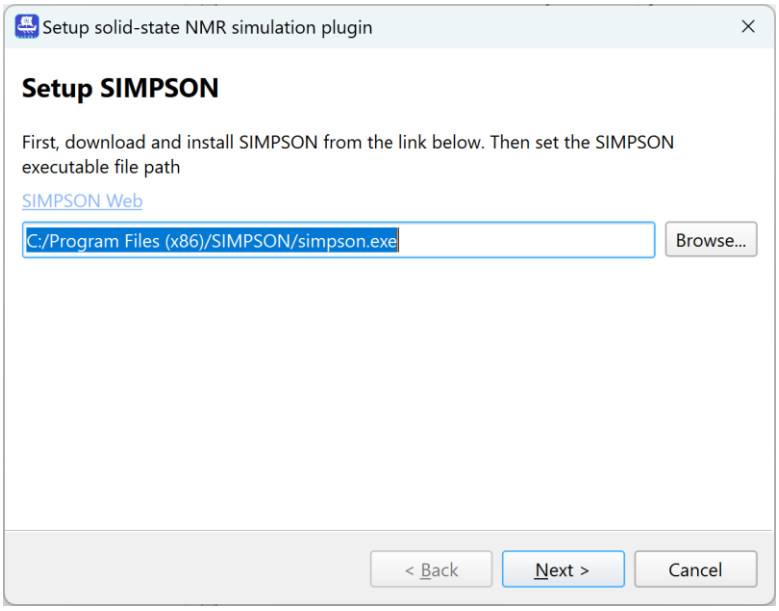
2.2 Configuration

SIMPSON is open-source software, developed outside JEOL. . JEOL does not distribute or support SIMPSON. Users must install SIMPSON separately and configure the JASON SolidSpin plugin according to these installation instructions. The plugin provides a wizard to guide the user through the necessary configuration steps.

If plugin is installed correctly, in JASON main menu (three dots) you should find a new entry "**New -> SolidSpin Simulation/Fit**":



Selecting this menu entry for the first time will initiate the Wizard:



First we would need to point the plugin to the location of SIMPSON executables. If SIMPSON is not installed, then clicking on the link SIMPSON Web will open the webpage where SIMPSON can be downloaded. Please, follow the installation instructions from the page. The link is correct at the time of creation of this manual.

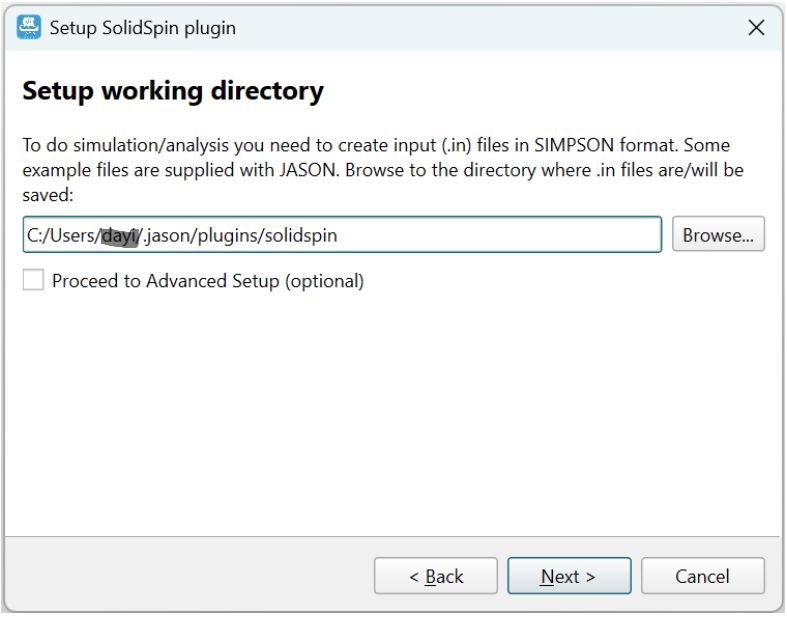
After SIMPSON is installed, the wizard field should be filled with the location of the executable SIMPSON file (*simpson.exe* on *Windows*). The **Browse** button can be used to browse the file system to locate the executable if the location is not determined automatically.

Click **Next** to proceed to the next stage.

At this stage we would need to set up a “working directory”. This directory will contain:

- SIMPSON input files that describe the simulations
- Temporary output files, unless otherwise specified

The default location points to the directory where the plugin libraries are installed. It also contains some example scripts (.in files).



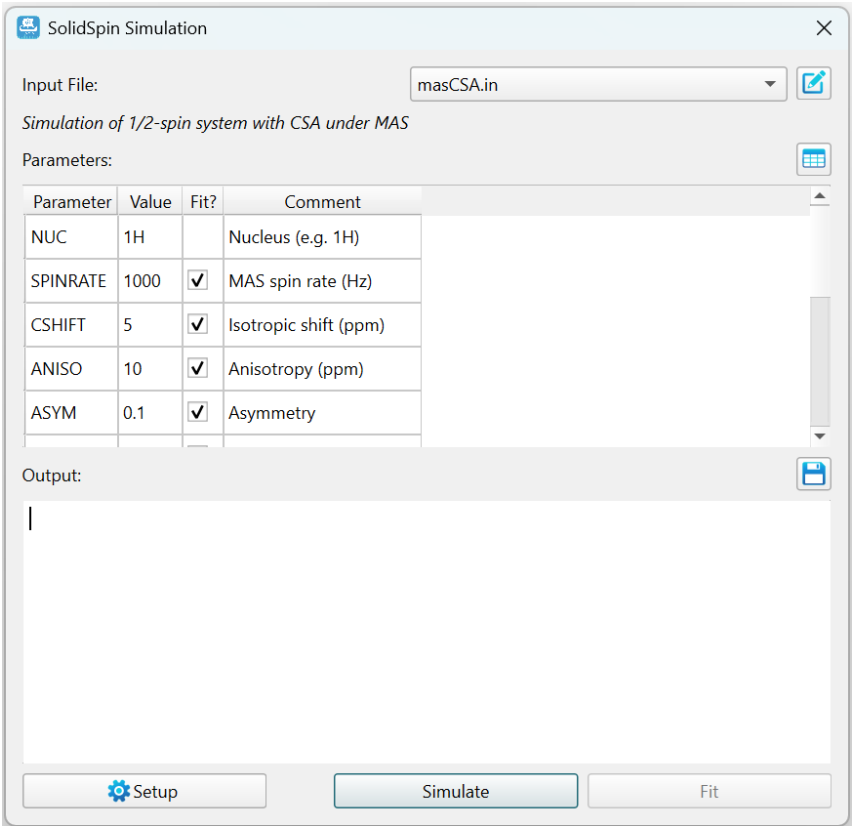
It is recommended to create a new directory and copy the examples to this new directory. Note that the directory must have read/write access.

You can check the box **Proceed to Advanced Setup** if special setup is needed and click **Next**. The advanced setup is not required for typical use and allows specify special command-line arguments for the SIMPSON executable to be supplied .

The next page of the Wizard should confirm the successful execution of the configuration steps.

Clicking “**New -> SolidSpin Simulation/Fit**” should now open the **Simulation** dialog.

The Setup Wizard can be restarted any time by clicking the **Setup** button in the left-bottom corner of the **Simulation** dialog.

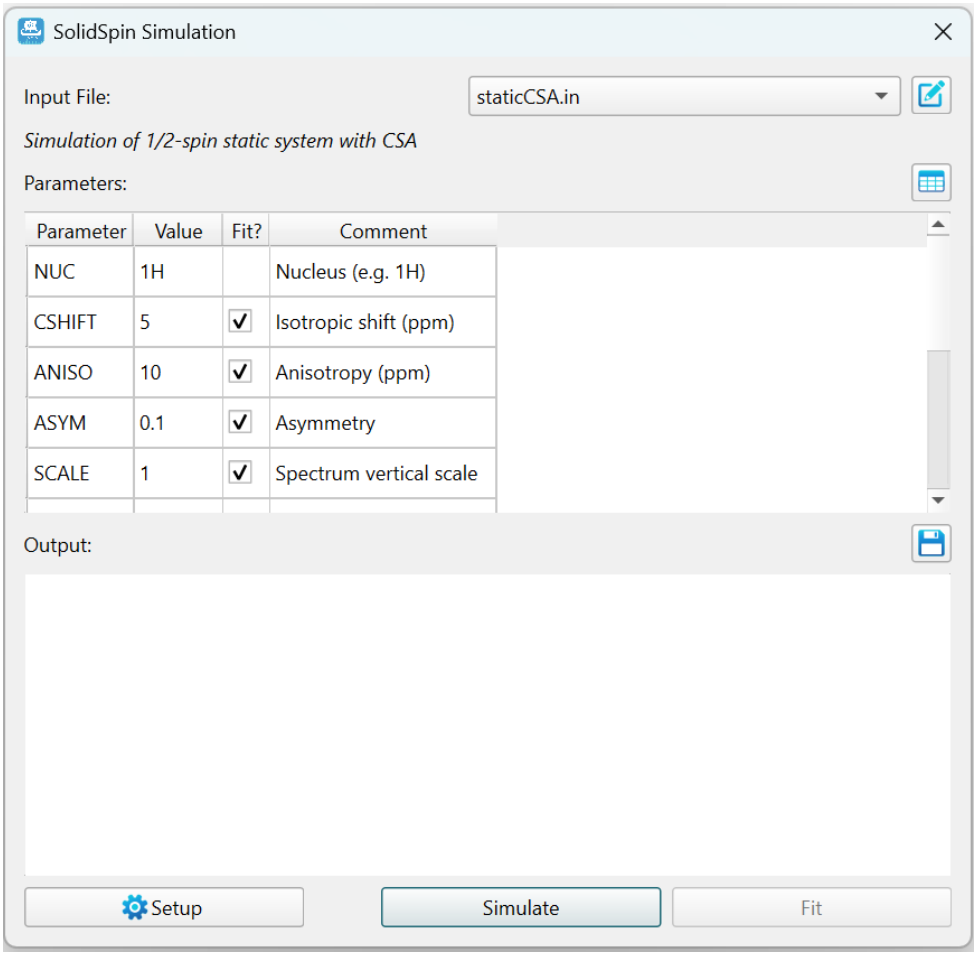


3 Static CSA simulation and fitting

3.1 Simple simulation

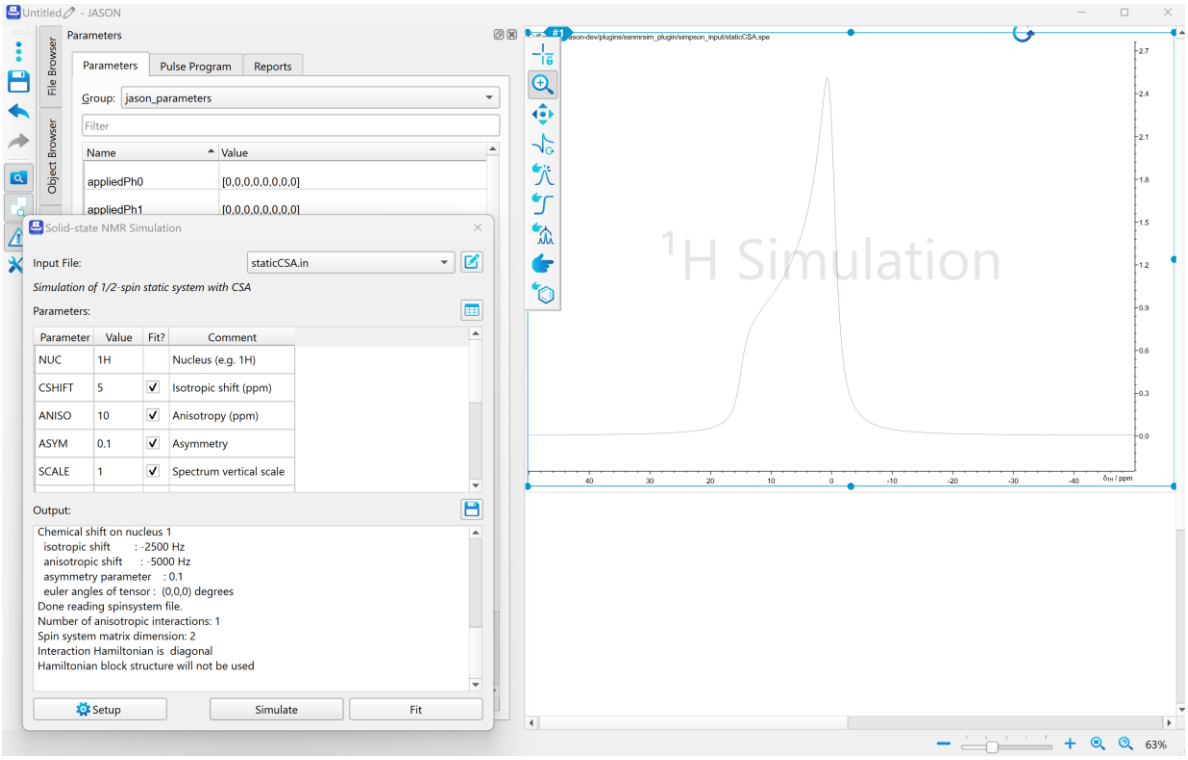
The first example we will discuss is a simple simulation of a CSA powder pattern.

- Open **Simulation** dialog using main menu (…) **New -> SolidSpin Simulation/Fit**
- Choose **Input file:** *staticCSA.in* from the drop-down menu. If you do not see the file in the drop-down menu, rerun the Setup Wizard and point the plugin to the location of the supplied example input files.
- The input file can be viewed or edited by pressing **Edit Input File** button next to the **Input File** drop-down menu. You will be asked to open the file in your favorite text editor. The dialog will be automatically updated when the input file is closed in the text editor.
- When an **Input File** is selected, the dialog will update showing the parameters specified in the *staticCSA.in* file:



The Parameters table contains user-provided parameters for the simulation. The **Value** column in the table is editable. Double-click on the value and enter the required parameters. The **Fit?** column is used for the fitting of experimental spectra and is not needed for a simple simulation. Its use will explained later in this manual.

Click **Simulate** button. The **Output:** field should be filled with the command-line output from SIMPSON (including possible execution errors). When simulation is finished, a simulated spectrum should be loaded to JASON canvas as a new item:



The Simulation dialog can be closed now by clicking **X** in the upper-right window corner.

3.2 Simulation, based on the experimental spectrum

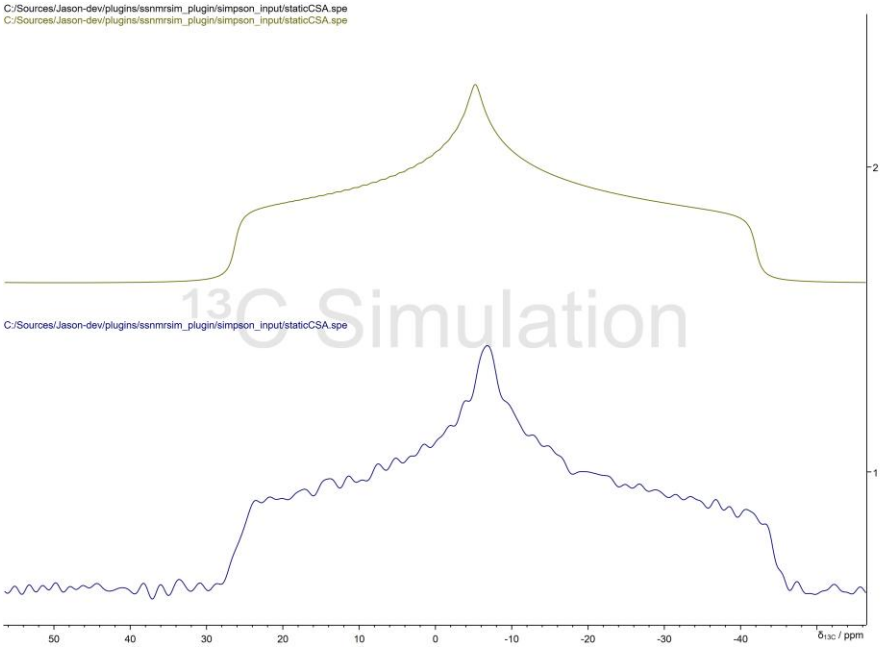
A more typical case of solid-state NMR simulation involves a comparison of a simulated spectrum with an experimental one. On this section we will show how it could be done in JASON.

- Open an example file *gly_static.spe*
- Open **Simulation** dialog using main menu (…) **New -> SolidSpin Simulation/Fit** while keeping experimental spectrum active on the canvas. The active spectrum is highlighted with the turquoise frame around it on the canvas. Click on the spectrum if it is not active.
- Select *staticCSA.in* input file from the drop-down menu
- Note, that the simulation plugin recognizes our intention to compare experimental and simulated spectra and fills some of the parameters in the table, matching them with the experimental parameters. This includes parameters: NUC, FREQ, SQ, NP, REF. Note that example spectra are not referenced and only supplied for illustration purposes.
- Modify the following parameters:

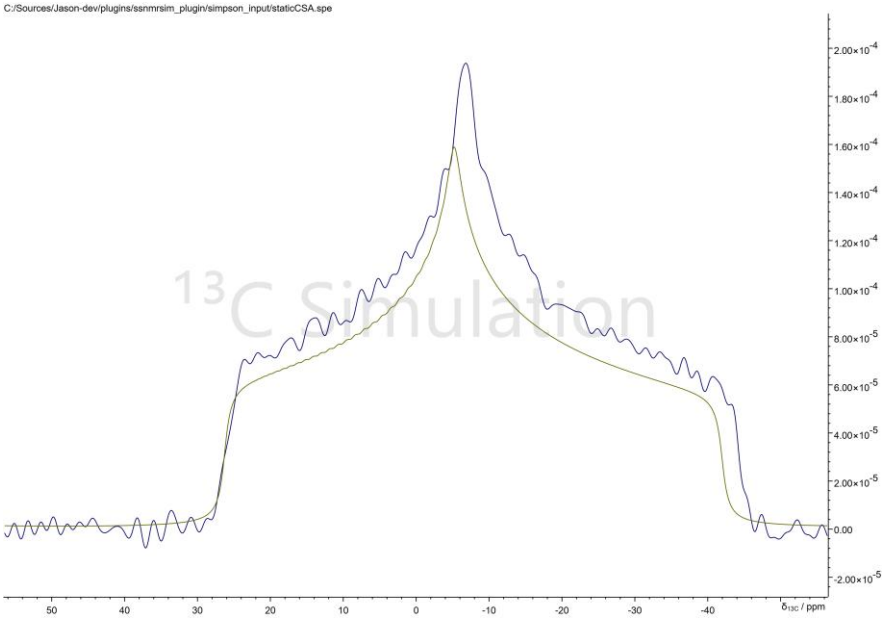
CSHIFT	-7
ANISO	-35
ASYM	0.9

SCALE	2e-4
CRYSFILE	zcw28656
LB	200

- Click **Simulate** to start simulation
- When simulation is finished, the simulated spectrum is loaded and appears as a vertical stack with the experimental spectrum:



- As an alternative, the spectra could be displayed, for example, in the overlapped mode by selecting **Spectra Overlap** from the drop-down menu in the JASON **Stack** panel. See the JASON manual on more details about the **Stack** panel:



You can see that we chose parameters reasonably close to the experimental ones but do not match exactly. The next section will discuss how to fit the experimental spectrum.

3.3 Fitting the experimental spectrum.

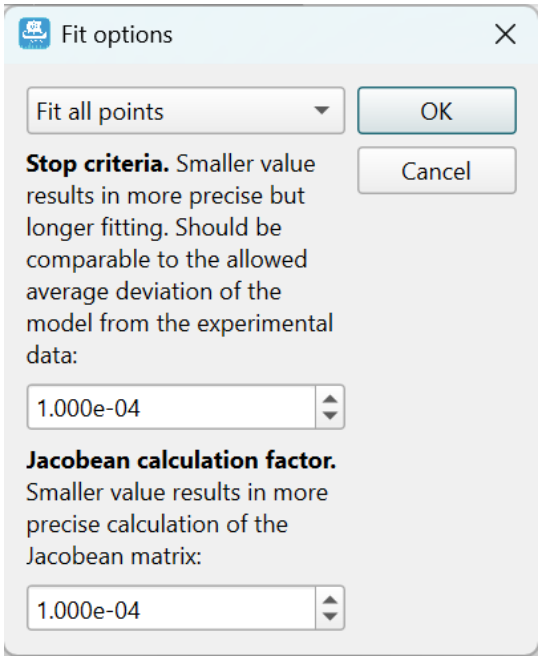
SolidSpin plugin uses JASON's non-linear fitting routine to automatically adjust specified parameters till the simulation fits the experimental spectrum. The non-linear fitting routine is sensitive to the initial value of the parameters. Selecting the parameters close to the true values helps to avoid local minima and speed up the fitting.

Execute the instructions from the previous section 3.2, setting up the initial parameters. It is recommended to adjust the parameters manually and run **Simulate** till simulated spectrum is close enough to the experimental spectrum.

Select the parameters that should be fitted. Note, that non-linear fitting benefits significantly from the reduction of the number of fit parameters. Please, consider carefully what parameters should be adjusted. Check the boxes in the **Fit?** column against the parameters that should be adjusted. The check box appears only against the parameters, specified to be suitable for fitting in the input file. We will discuss the format of the input file in the further section of this manual.

For the static CSA example, we can keep all possible parameters checked: CSHIFT, ANISO, ASYM, SCALE.

Click **Fit** to start fitting. A dialog with fitting parameters will appear:

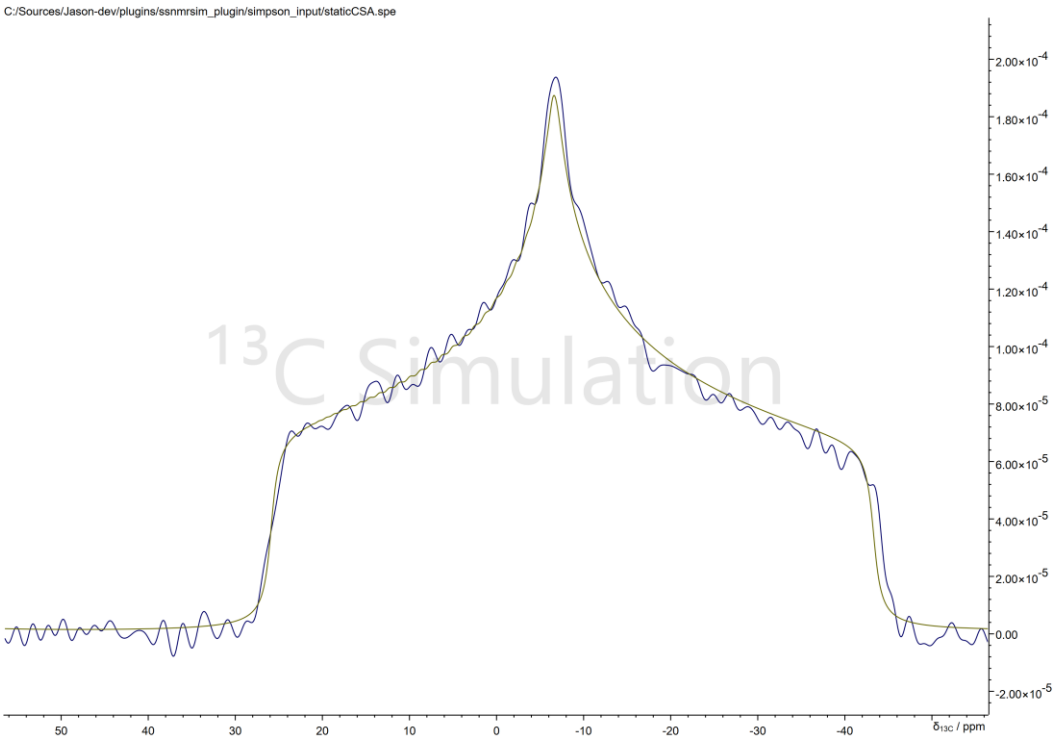


Ensure that the **Fit all points** option is selected. With this option all points of the experimental spectrum will be compared with the simulation during the fitting process. We will discuss other options from the drop-down menu later.

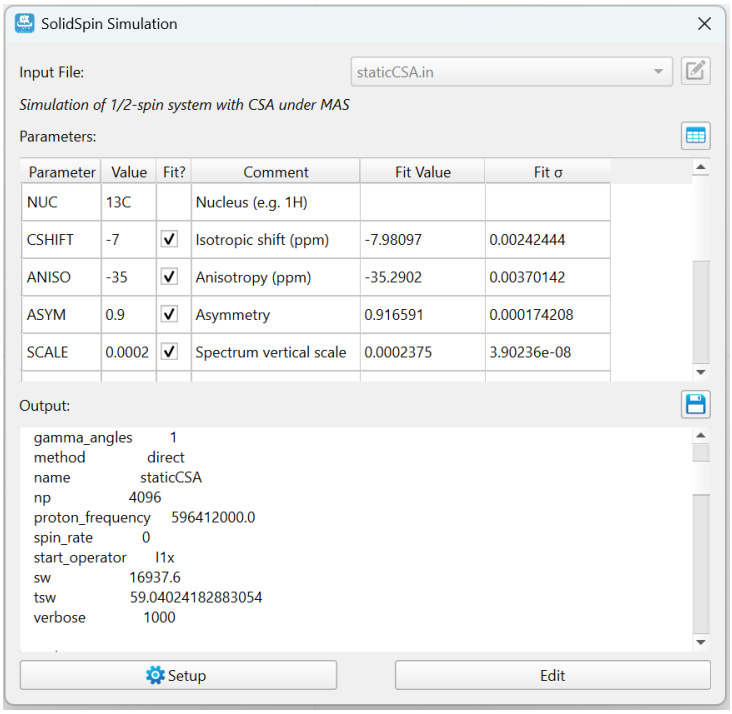
The other two parameters specify the quality of the fit.

Click **OK** and wait till fitting is finished, whichcould take a few minutes. During the routine the simulated spectra will be updated in the canvas item, showing the progress of the fitting. If needed, the fitting routine can be interrupted by clicking the **Exit Simulation** button.

A dialog will appear when the fitting is successfully finished. The final spectrum is displayed in the canvas item:



The **Parameters** table has got updated with additional columns **Fit Value** and **Fit σ** containing the optimized value and its error:



Note the table is in read-only mode after the fitting. If you wish to redo the fitting, you will need to press **Edit** button to reactivate the editing mode. The fit columns will be removed. You have the option to either keep the original initial parameters or transfer the last fitted values as the initial parameters for the next cycle of fitting.

To report the results of the analysis, the Parameters table could be transferred to the canvas by pressing **Create a table on the canvas** button.

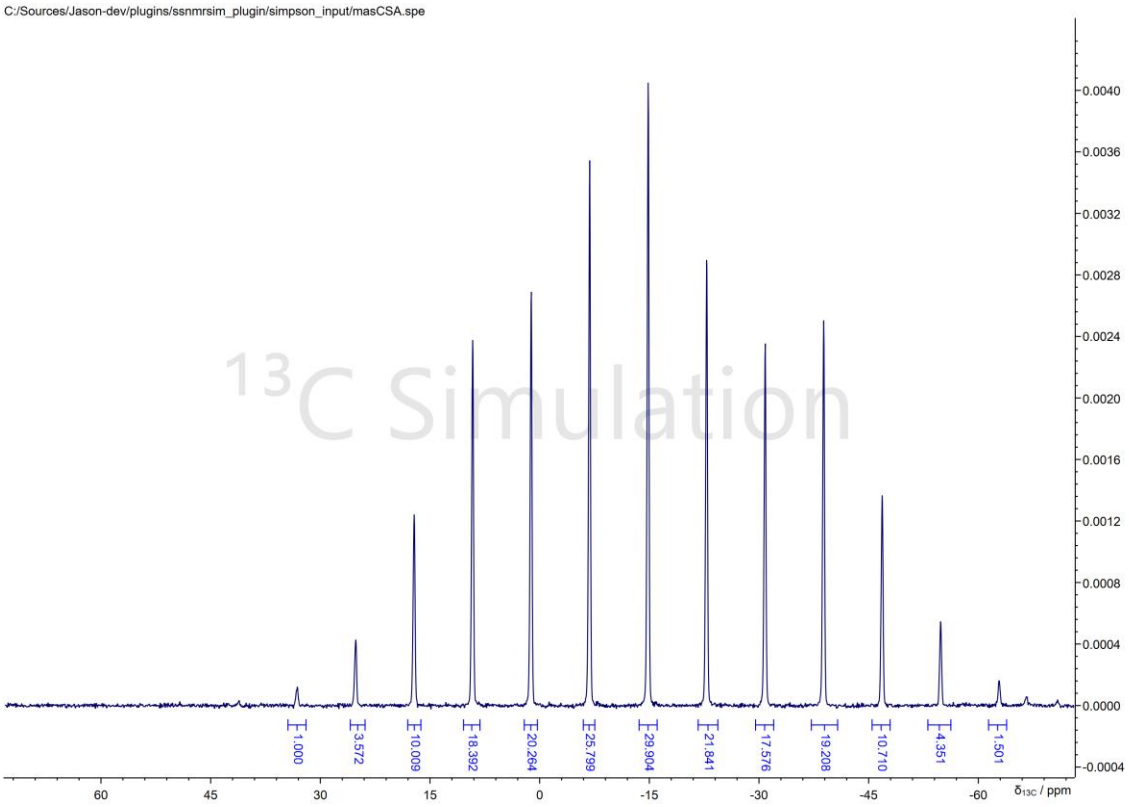
4 MAS CSA simulation and fitting

In this example we will fit MAS sidebands pattern for spin-1/2 spectrum with CSA. Many steps are similar to the static CSA simulation. So, we will focus on specific steps and functionality, important for this example.

- Open an example file *13C_gly_CP_1kHz.spe*

Though a fitting using all spectra points is possible, in practice the sharp features of the spectrum can make fitting unstable with many possible local optimization minima. Therefore, we will use a special approach when only the overall sideband intensities are used for fitting. For that we need to make some preparations.

Select the **Manual integrals** tool in JASON and integrate the regions around each sideband. For better representation, the **Integral Curve** could be switched off in the **View** panel. The integrated spectrum should look like this:



The value of each integral corresponds to the overall intensity of each spinning sideband. Only those absolute integral values (13 values on the screenshot) will be used in the fitting routine. Note, that this way is also possible to select a sidebands pattern in more complex spectra with several NMR resonances.

- Open **Simulation** dialog using main menu (…) **New -> SolidSpin Simulation/Fit** while keeping experimental spectrum active on the canvas.
- Select *masCSA.in* input file from the drop-down menu
- Modify the following parameters:

CSHIFT	-14.85
ANISO	-35

ASYM	0.9
SCALE	2e-4
LB	28

- Click **Simulate** to start simulation and check that simulated pattern approximates sufficiently the experimental spectrum.

Let's consider the parameters which need to be included in fitting as we are going to use the integral values.

As we use the integrals, the small deviations in CSHIFT (isotropic chemical shift) and SPINRATE (MAS rate) will not have significant impact on fitting as far as the simulated sidebands are close enough to experimental ones. So, it's better to estimate those values from the experimental spectrum and fix them (by unchecking them in **Fit?** column) during the fitting.

The LB (line broadening) and SCALE (spectrum intensity scaling) are also strongly anti-correlated regarding the integral values. The same absolute integral can be obtained, e.g. by decreasing LB and increasing SCALE to some degree. Such correlated parameters are undesirable and can destabilize the fitting routine. So, we will fix LB value in the fitting as well. Luckily, it's relatively easy to estimate those values from the experimental data. The exact match is not critical for our purposes.

Thus, the fitting will be performed on the parameters: ANISO, ASYM and SCALE only.

- Click **Fit**
- In the appearing dialog select **Fit integral values** to fit the integrals only, as discussed above.
- It could be beneficial to adjust the fitting parameters, e.g. by setting them to 1e-6 each.
- Run fitting.

Keep in mind that only the CSA parameters are adjusted so far. If a better-quality fit is required, as a next stage the experimental spectrum points could be used to fit skipped parameters: CSHIFT, SPINRATE and LB. The option **Fit all points** or **Fit within integral ranges** can be used. The option **Fit within integral ranges** uses the user-defined integrals. But instead of using the single integral value as in **Fit integral values**, it uses all spectral points within integral ranges.

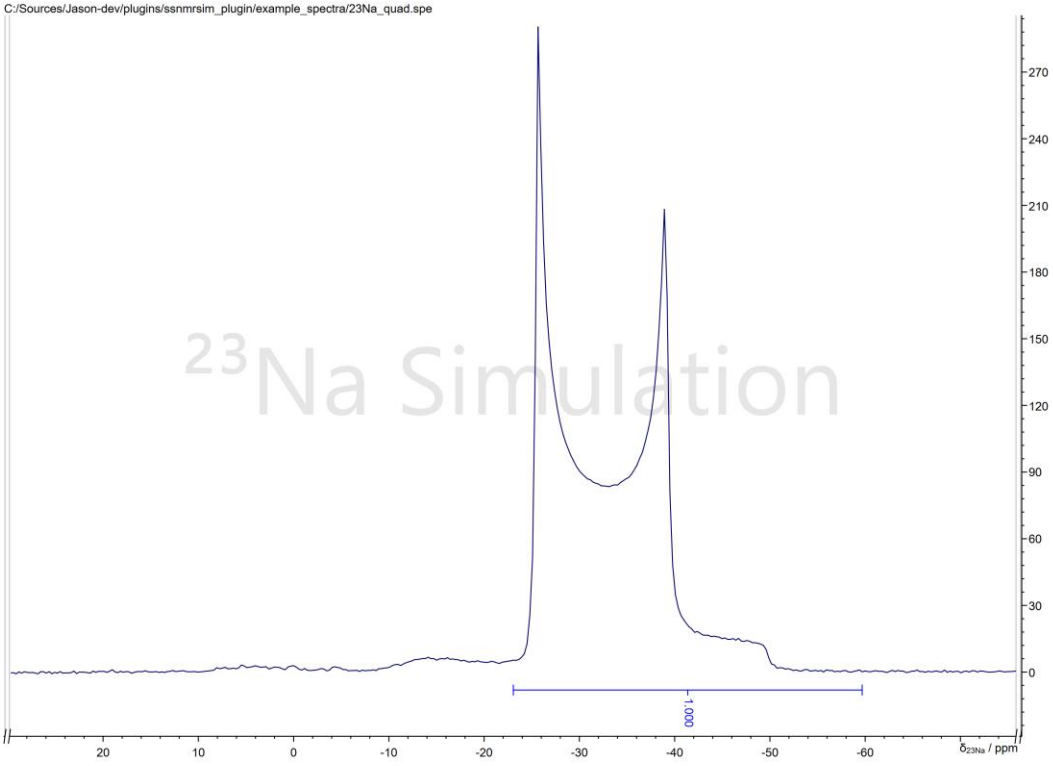
5 Fitting spectra of spin-N/2 quadrupolar spectra with second-order quadrupolar broadening.

In this example we will fit the central transition of the MAS spectrum of ^{23}Na, affected by the second-order quadrupolar broadening. Many steps are similar to the previous simulations. So, we will focus on specific steps and functionality, important for this example. We will use **Fit within integral ranges** in this section.

- Open an example file *23Na_quad.spe*.

This spectrum has a broad spectrum window. So, we will do two preparatory steps:

- To specify the fitting range, select an integral around the central transition as shown below:



Note, that at the left integrals exclude some undesirable resonances which will not be reproducible in simulations and can affect the fitting results if included in the fitted range. On the right the integral is extended a bit to include the baseline.

On the screenshot above the **Manual cuts** were also applied on the experimental spectrum. The cuts hide the left and right parts of spectrum as indicated by the cut markers (//) on the horizontal axis. This is only a visual adjustment that allows to see only the important part of the spectrum during the fitting. It doesn't alter the spectrum points.

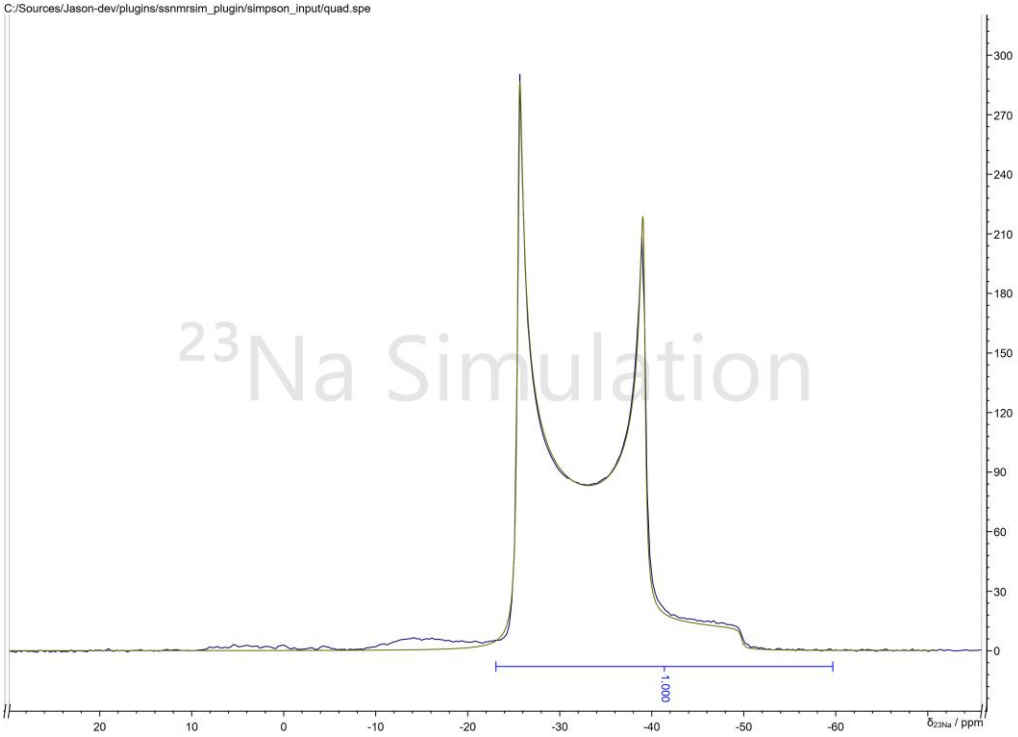
- Select *quad.in* input file from the drop-down menu
- Modify the following parameters:

CSHIFT	-22
ORDER	2
CQ	2.2
ETA	0.0

SCALE	10
LB	20
SW	20000

Note that the simulated spectrum will have narrower spectral window than the experimental spectrum. In general, the sampling of the simulated spectrum does not need to match the sampling of the experimental spectrum. Only the common range of two spectra will be used for the target function in fitting. The simulated spectrum will be resampled using interpolation to match the points of the experimental spectrum.

- Run **Simulate** to check that simulated pattern approximates sufficiently the experimental spectrum.
- Set the fitting parameters: CSHIFT, CQ, SCALE, LB by checking the boxes in the **Fit?** column. As we fit a narrow range of the spectrum, the SPINRATE will be insensitive and should be set as a fixed parameter approximately equal to the experimental value. It's also easy to see that ETA could be set as fixed to 0.0 for this spectrum though it could be fitted as well.
- Click **Fit**.
- Select **Fit within integral ranges**. As previously we only specified a single integral, its range will be used for the fitting. That excludes any undesirable resonances from the fitting routine. The default (1e-4) values for the fitting parameters could be kept.
- The result of the fitting should be like the one below:



To report the results of the analysis, the Parameters table could be transferred to the canvas by pressing **Create a table on the canvas** button.

6 Input file format.

The full power and flexibility of the SIMPSON simulations can be used by creating custom input files and adding them to the simulation library.

The files are standard *.in* SIMPSON files with some modifications. The description of the *.in* format is outside the scope of this manual and it was discussed (with examples) in the publications from the authors of SIMPSON. For example:

https://doi.org/10.1006/jmre.2000.2179

http://dx.doi.org/10.1016/j.jmr.2014.07.002

https://doi.org/10.1016/bs.arnmr.2019.12.001

We will focus only on the modifications required by the JASON SolidSpin plugin.

This modification allows us to define the graphical user interface (GUI) for the SolidSpin **Simulation** dialog as well as facilitate the exchange of parameters between SIMPSON and the JASON plugin.

The GUI is defined in a series of custom strings within *.in* file which appear as comments for SIMPSON and ignored during the simulation. The JASON plugin, instead, parses these lines and takes the simulation title and parameter values from them.

SIMPSON and JASON exchanges parameters via temporary system variables. When simulation is requested, the SolidSpin plugin creates several temporary system variables (those are present during the simulation only) and runs SIMPSON. The SIMPSON *.in* file should have a code to extract the values of these system variables and use them in simulation.

The fitting routine uses the internal JASON fitting engine. The JASON plugin will run multiple simulations with the trial parameter values. This approach simplifies the *.in* files as no fitting code should be added there. A simple single spectrum simulation input can be used.

Let's discuss the *masCSA.in* as an example*:*

```
##title:<i>Simulation of 1/2-spin system with CSA under MAS</i>

##par:NUC;1H;;fixed;Nucleus (e.g. 1H)
##par:SPINRATE;1000.0;10.0,1000000.0,1;variable;MAS spin rate (Hz)
##par:CSHIFT;5.0;-100000.0,100000.0,1;variable;Isotropic shift (ppm)
##par:ANISO;10.0;-10000000.0,10000000.0,1.0;variable;Anisotropy (ppm)
##par:ASYM;0.1;0.0,1.0,0.05;variable;Asymmetry
##par:SCALE;1.0;0.0,100000000000000.0,0.001;variable;Spectrum vertical scale
##par:LB;50;0.0,10000.0,1.0;variable;Linebroadening (Hz)
##par:FREQ;500.0;1.0,3000.0,100.0;fixed;Proton frequency (MHz)
##par:CRYSFILE;rep320;;fixed;Crystal file
##par:SW;50000;10.0,20000000.0,100.0;fixed;Spectrum width (Hz)
```

```
##par:NP;8192;1024,524288,1024;fixed;Number of points in FID
##par:REF;0;-20000000.0,20000000.0,10.0;fixed;Reference shift (Hz)

spinsys {
  channels 1H
  nuclei 1H
  shift 1 0 10p 0 0 0 0
}

par {
  spin_rate       -1

  np             2048
  proton_frequency -1
  start_operator   I1x
  detect_operator  I1p
  method          direct
  crystal filé     -1
  gamma_angles     8
  sw              -1
  variable tsw     -1
  verbose         1111
}

proc pulseq {} {
  global par

  acq_block {
    delay $par(tsw)
  }
}

proc main {} {
  global par
  global spinsys
  set spinsys(channels) $::env(NUC)
  set spinsys(nuclei) $::env(NUC)
  #spectrometer frequency for the isotope NUC
  set sfrq [resfreq $::env(NUC) [expr $::env(FREQ)*1e6]]
  #Isotropic shift in Hz with the adjustment of spectrum reference
  set adjshft [expr $::env(CSHIFT)*($sfrq+$::env(REF))*1e-6-$::env(REF)]
  set spinsys(shift) "1 $adjshft $::env(ANISO)p $::env(ASYM) 0 0 0"

  set par(spin_rate) $::env(SPINRATE)
  set par(proton_frequency) [expr $::env(FREQ)*1e6]
  set par(crystal_file) $::env(CRYSFILE)
  set par(sw) [expr {floor($::env(SW)/$par(spin_rate))*$par(spin_rate)}]
  set par(tsw) [expr {1e6/$par(sw)}]

  set f [fsimpson]
  fset $f -ref $::env(REF)
  faddlb $f $::env(LB) 0
```

```
  fzerofill $f  $::env(NP)
  fft $f
  fphase $f -scale $::env(SCALE)
  fsave $f $par(name).spe
}
```

6.1 GUI definitions

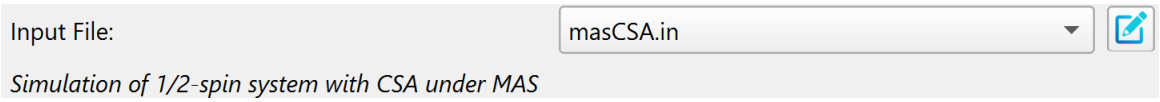
The file starts with a custom section which defines the graphical user interface, presented to the user. Each line of this section starts with "##". The # is used as a comment's identifier in SIMPSON. So, during the execution SIMPSON ignores this section. We use double # to separate comments from the GUI definitions.

The example line:

##title:Simulation of 1/2-spin system with CSA under MAS
has a following format:

##title:<user-defined title for the simulation>.

The title will appear in the Simulation dialog when the simulation file is selected. Any text can be used to explain the functionality of the input file. HTML tags are allowed. In the example above the title will be shown in italic using … HTML tags:



The lines that start with ##par: specifies rows in the **Parameters** table and have the format:

##par:<parameter name>;<default value>;<minimum>,<maximum>,<step>;<variable/fixed>;<comment>

- <parameter name>: the name of the parameter and, also the name of the system variables. As discussed above SIMPSON and JASON simulation plugin exchanges the parameters values via temporary system variables. Therefore, the restrictions for the parameter names are the same as the restrictions for the system variables names. For simplicity, a single word, all capitals characters without special characters are recommended to use for the name.
- <value> is the default value which appears in the SolidSpin **Simulation** dialog. Could be a number or a string. If simulation starts with an active experimental spectrum, the values of some reserved parameters will be taken from the experimental spectrum and used instead of the default value. Those parameters are:
    - NUC: nucleus, e.g. 13C
    - SW: spectral window in Hz
    - FREQ: spectrometer frequency in MHz
    - REF: referencing shift in Hz. This is a value in Hz associated with the center of the spectrum
    - SPINRATE: spinning rate, e.g. MAS rate in Hz
    - NP: the number of spectral points

- <minimum>, <maximum>, <step> specifies the allowed minimum and maximum values for the parameter. <step> is used as the step when up/down arrows are used in GUI to change the value. The <minimum> and <maximum> are used also as the parameter constraints in fitting. The parameter will be restricted withing this range during the fitting. Note, that the <minimum>, <maximum>, <step> are separated by ",", while main sections of the ##par string are separated by ";". The <minimum>, <maximum>, <step> can be omitted, e.g. for the string parameters.
- <variable/fixed>: if "variable" keyword is used for the parameters, the parameter can be adjusted during the fitting. A checkbox will appear in the **Fit?** column for this parameter and user can choose to either fit this parameter or keep it constant during the fitting. All parameters, marked as "fixed" are always kept constant.
- <comment> is a user-defined string which explains the use of the parameter.

Examples:

##par:SPINRATE;1000.0;10.0,1000000.0,1;variable;MAS spin rate (Hz)

specifies a parameter and system variable SPINRATE. The default value is 1000.0. As it is one of the reserved parameter names, its value could be replaced with the experimental value of the MAS spin rate, if the active experimental spectrum is provided for the simulation. The allowed values should be in the range of 10.0-1000000.0. GUI increment step of 1.0 will be used. The parameter can be fitted as it's specified as *variable*. A string *"MAS spin rate (Hz)"* will be displayed in the **Comment** column of the **Parameters** table.

##par:CRYSFILE;rep320;;fixed;Crystal file

specifies a parameter and system variable CRYSFILE. The default value is a string "rep320". This is one of the names of "crystal files" defined in SIMPSON. The allowed values range and GUI step is not applicable for any string parameters and omitted. The parameter is *fixed*. Of course, only numerical parameters could be defined as variable for fitting. Only "fixed" is allowed for string parameters. A string *"Crystal file"* will be displayed in the **Comment** column of the **Parameters** table.

6.2 Sections spinsys, par and proc pulseq

These sections remain mostly unmodified from the standard SIMPSON files. It should be noted that the values of many parameters listed in those sections, may need to be replaced by the values, passed from the JASON plugin via system variables. In our examples it's done later in the *proc main* section. The *proc main* is Tcl code which will be executed when the simulation is called and it's convenient to set the value of parameters at this point.

For readability purposes, it could be recommended to mark the values in the *spinsys* and *par* that would be modified later. In the example above, it's done by setting it to '-1'. For example:

```
crystal_file    -1
```

or

```
sw              -1
variable tsw    -1
```

6.3 Section proc main

As discussed above, this section has some modifications, specific for the use of the plugin. The *proc main* is Tcl code which will be executed when the simulation is called. It is convenient at this point to set the values of parameters passed from the JASON plugin via system variables.

The Tcl-language syntax for accessing system variables is:

$::env(<variable_name>)

You can find some examples at the beginning of the *main* section. For example:

```
set spinsys(nuclei) $::env(NUC)
...
set par(spin_rate) $::env(SPINRATE)
set par(proton_frequency) [expr $::env(FREQ)*1e6]
```

The last line also demonstrates how the Tcl command *expr* can be used to do calculations. In this case it converts the user-provided frequency expressed in MHz to the Hz value, required by SIMPSON.

Let's discuss some special cases.

The example uses a special calculation for the spectral window:

```
set par(sw) [expr {floor($::env(SW)/$par(spin_rate))*$par(spin_rate)}]
```

To keep numerical simulations efficient, it is required sometimes to restrict the values of some simulation parameters. For example, it's highly beneficial to keep acquisition dwell time (and thus the spectral window) to have a common synchronization factor (an integer multiplier between them) with the MAS rotor period. But, if fitting of an experimental spectrum is performed, the spectrum window of the experimental spectrum may not satisfy this condition.

The SolidSpin plugin overcomes this problem by removing the requirement of the same sampling in the experimental and simulated spectra. The resampling of the simulated spectrum will be applied if it doesn't match the sampling of the experimental spectrum. Only the common range of two spectra will be used for the resampling. A linear interpolation within this region will be applied to find the simulated points at the exact positions of the experimental points. Those resampled simulated points will be used in the fitting routine.

This way the only sampling requirements for the simulated spectrum is that it should cover the region of interest inside the experimental spectrum and digitization of the simulated spectrum is sufficient so the interpolation can be used in the fitting routine.

In the code above the *par(sw)*, used in simulation, is estimated from the experimental spectral window as *$::env(SW)* but adjusted to be the closest multiple of the MAS spin rate. This will ensure that simulation is performed with high numerical efficiency, regardless of the exact experimental values of the spectrum window and the MAS spin

rate. E.g. optimizing MAS spin rate is possible this way as well. Interpolation will take care of the small deviations between the experimental and simulated spectral windows.

Another special consideration is made about the isotropic chemical shift values and spectrum referencing. The related code is:

```
#Isotropic shift in Hz with the adjustment of spectrum reference
set adjshft [expr $::env(CSHIFT)*($sfrq+$::env(REF))*1e-6-$::env(REF)]
set spinsys(shift) "1 $adjshft $::env(ANISO)p $::env(ASYM) 0 0 0"
```

SIMPSON simulated spectra are referenced after the simulation in this command from the *main* section:

```
fset $f -ref $::env(REF)
```

The REF parameter above is set to the position of the center of the spectrum, expressed in Hz. It will be estimated and set in the simulation dialog automatically from the experimental spectrum if the simulation is started to compare with an experimental spectrum.

However, the referencing is set **after** the simulation is completed in SIMPSON. The isotropic chemical shift, used in *spinsys(shift)* entry are not referenced (*REF=0.0*). This is inconvenient: those values do not correspond to the chemical shifts observed on the experimental spectrum.

So, the extra code is used to recalculate the referenced chemical shift values from the experimental spectrum to the non-referenced values, used by the simulation:

```
set adjshft [expr $::env(CSHIFT)*($sfrq+$::env(REF))*1e-6-$::env(REF)]
```

The variable *adjshift* (adjusted shift) contains the adjusted shift, expressed in Hz.

First, the referenced chemical shift is converted to Hz:

```
$::env(CSHIFT)*($sfrq+$::env(REF))*1e-6
```

using *$sfrq+$::env(REF)* (both variables are in Hz) as a reference frequency.

Then it is shifted by *$::env(REF):*

$::env(CSHIFT)*($sfrq+$::env(REF))*1e-6-$::env(REF).

The computed value is used for the simulation without ‘p’ prefix as it is expressed in units of Hz:

```
set spinsys(shift) "1 $adjshft $::env(ANISO)p $::env(ASYM) 0 0 0"
```