Jellyfish manual

Wouter Franssen & Bas van Meerten

23rd December 2019

Contents

1 Introduction

Jellyfish is a program for the simulation of 1D NMR spectra for liquid state samples specialised in complicate J-coupling patterns. It features a graphical user interface for intuitive simulations, and a python library for advanced use. Jellyfish is written in the Python programming language, and is cross-platform and open-source (GPL3 licence).

The Jellyfish graphical interface is aimed for teaching purposes, especially when it comes to demonstrating the influence of strong coupling effects on NMR spectra in the liquid state. The engine itself is quite fast, as several more advanced techniques for simulating J-coupling patterns are implemented. These are discussed in the technical mart of this manual. Note that Jellyfish calculations are exact: no approximations are made to reduce the calculation time.

2 Running Jellyfish

2.1 Python and library versions

Jellyfish has been programmed to run on both the python 2.x and 3.x. Jellyfish should run on python versions starting from 2.7 and 3.4. For the library version, the following are needed:

- `numpy` >= 1.11.0
- `matplotlib` >= 1.4.2
- `scipy` >= 0.14.1

Jellyfish also needs the `PyQt` (version 4 or 5) library.

2.2 Installing

2.2.1 Linux

On Linux, Jellyfish can be most efficiently run using the python libraries that are in the repositories. For Ubuntu, these can be installed by running:

```
sudo apt install python python-numpy python-matplotlib python-scipy
python-pyqt5
```

Navigating to the `Jellyfish` directory in a terminal, Jellyfish can then be run by executing `python Jellyfish.py`.

2.2.2 Windows

On Windows, the relevant python libraries can be installed by installing `anaconda`: `https://www.anaconda.com/download/`. If you do not have another python version installed already, make sure to add anaconda to the `path` during the install. In this case, Jellyfish can be run by executing the `WindowsRun.bat` file in the `Jellyfish` directory. Desktop and start menu shortcuts to this file can be created by executing the `WindowsInstall.vbs` file.

If you already have other version of python installed, adding anaconda to the `path` might create issues. Do not do this in this case. When not added to the path, the `WindowsRun.bat` should be edited in such a way that `pythonw` is replaced with the path to your `pythonw.exe` executable in the anaconda install directory.

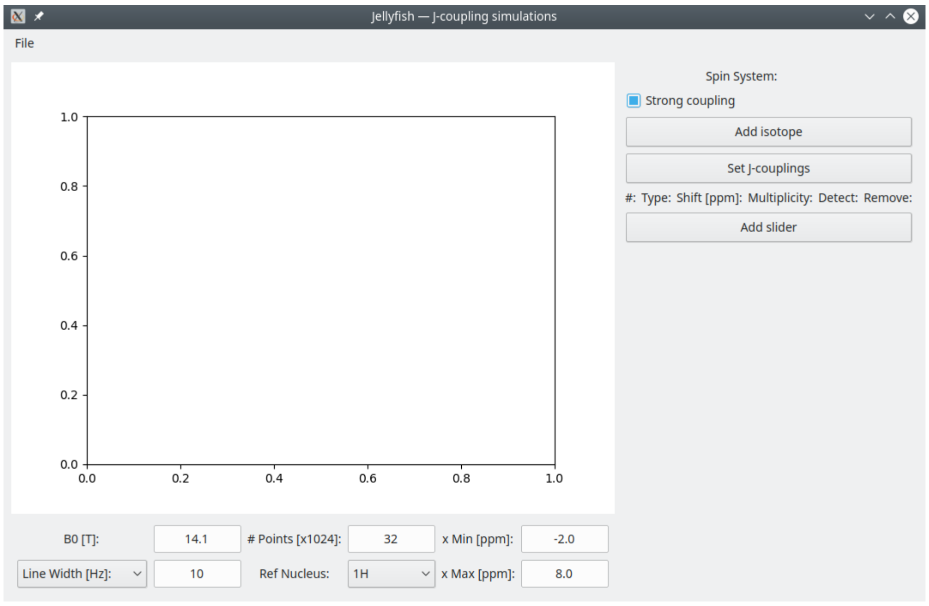
2.2.3 OS X

On OS X, the relevant python libraries can be installed by installing `anaconda`: `https://www.anaconda.com/download/`. Navigating to the `Jellyfish` directory in a terminal, Jellyfish can then be run by `anacondapython Jellyfish.py`, with `anacondapython` replaced by the path to your anaconda python executable.

3 The graphical user interface

Jellyfish can be run either as a standalone program, via its graphical user interface (GUI), or as a library by loading it in a Python script. Firstly, I will describe how to use the GUI.

Opening the program shows the main menu:



Strong coupling

B0 [T]: 14.1

Points [x1024]: 32

x Min [ppm]: -2.0

Line Width [Hz]: 10

Ref Nucleus: 1H

x Max [ppm]: 8.0

The window has two regions of settings: the spin system on the right, and the spectrum settings in the bottom. Also, there is a plot region.

3.1 Plot region

There are several ways to control the display of the spectrum using the mouse. Below, a list of these is given:

- Dragging a box while holding the left mouse button creates a zoombox, and will zoom to that region.
- Dragging while holding the right mouse button drags (i.e. pans) the spectrum. Doing this while holding the Control button pans only the x-axis, holding Shift pans only the y-axis.
- Double clicking with the right mouse button resets the view to fit the whole plot (both x and y).
- Scrolling with the mouse wheel zooms the y-axis. Doing this while also holding the right mouse button zooms the x-axis. By holding Control the zooming will use a larger step size.

3.2 The bottom frame

In the bottom frame, several parameters can be changed:

- B0 [T]: the magnetic field strength in Tesla. This value can also be changed from a slider (see below)
- Line Width [Hz/ppm]: the line broadening added to the spectrum simulation. Input either in Hertz or ppm (can be changed via the dropdown menu). The ppm setting is most useful when changing the magnetic field: the effective line width remains constant then.
- # Points [x1014]: The number of points in the spectrum.
- Ref Nucleus: Defines the reference frequency (i.e. nucleus) of the spectrum.
- x Min [ppm]: The minimum x-value in ppm
- x Max [ppm]: The maximum x-value in ppm

3.3 The spin system frame

On the right-hand side of the window, the spin system can be defined. Jellyfish support spin systems of any size (but your computer will not be able to handle very large systems, of course). There are several buttons always available:

- Strong coupling: toggle that determines if strong coupling is present. This determines if the J_{xy} part of the Hamiltonian is added. In realty, this is always on. The toggle is provide here to be able to show the effect of this strong coupling.
- Add isotope: Adds an isotope to the spin system. The gives a pup-up window were the type of nucleus can be selected, the chemical shift can be given, and the multiplicity defined. Pushing OK adds this spin to the system. This adds a row of input boxes to the interface, were the shift and multiplicity can be changed.
- Set J-couplings: Opens a window to set the J-coupling parameters. By default, all are 0. The amount of entries depends on the spin-system that has been defined at the moment.
- Add slider: Adds a slider to the interface. This can be used to change a specific parameter while watching the resulting spectrum. Ideal for teaching purposes on the effect of strong coupling, field strengths, etc. Pushing this button opens an input window, where the type (B0, Shift, or J-coupling) needs to be defined, as well as the minimum and maximum value of the slider. For the 'Shift' or 'J-coupling' type, one or two spin numbers also need to be given.

When a spin is defined, an extra line is added to the interface with the following options:

- Shift [ppm]: the chemical shift in ppm
- Multiplicity: the multiplicity of the nucleus (i.e. the ^{1}H multiplicity of a CH_3 group is '3')
- Detect: whether or not this nucleus is detected (i.e. added to the spectrum)

- X: remove this nucleus from the spin system. This also clears all relevant J-couplings, and removes and slider that has a relation to this spin.

When a slider is defined, it adds a row to the slider space. Here, there now is a slider which can be used to change the specified parameter. There is also a 'X' button to remove the slider. When using a slider, the respective value is also changed in the rest of the interface (e.g. chemical shift are also updated in the spin system rows).

3.4 Menu

Jellyfish also has a menu on the top of the interface. It has the following options:

- File:
  - Export Figure: export the current plot as an .png image.
  - Export Data (ASCII): save the current spectrum as a text file (x and y data).
  - Export as ssNake .mat: save spectrum as a Matlab file as used by the ssNake software (see `https://www.ru.nl/science/magneticresonance/software/ssnake/`).
  - Export as Simpson: save as a text file as supported by the Simpson simulation software (see `http://inano.au.dk/about/research-centers/nmr/software/simpson/`).
  - Quit: closes the program

4 Running as a script

Apart from running Jellyfish as a program via the user interface, it can also be used as a library from within python. This requires that you have the source code of Jellyfish (and not a compiled version). The 'Examples' directory holds some examples on how to do simulations from a script. Below an easy example is given. Here, it is assumed that the Jellyfish main directory (which holds `engine.py`) is one directory higher than this file.

```
import numpy as np
import sys
sys.path.append("..")
import engine as en

#---------Spectrum settings---------------
Base = 42.577469e6
RefFreq = 600e6 #zero frequency
B0 = RefFreq/Base #B0 is proton freq divided by the base scale
```

```
StrongCoupling = True #Strong coupling on
Lb = 0.2 #Linewidth in Hz
NumPoints = 1024*128 #Number of points
Limits = np.array([-1,3]) #Limits of the plot in ppm

#---------Spin system---------------
# add spin as ['Type',shift, multiplicity,Detect]
SpinList = [['1H',0,1,True]]
SpinList.append(['1H',2,1,True])

Jmatrix = np.array([[0, 10],
                    [ 0, 0]])



#---------Make spectrum------------
# Prepare spinsys:
spinSysList = en.expandSpinsys(SpinList,Jmatrix)
# Get frequencies and intensities
Freq, Int = en.getFreqInt(spinSysList, B0, StrongCoupling)
# Make spectrum
Spectrum, Axis, RefFreq = en.MakeSpectrum(Int, Freq,
                                    Limits, RefFreq,Lb,NumPoints)
#Save as ssNake Matlab file
en.saveMatlabFile(Spectrum,Limits,RefFreq,Axis,'easy.mat')
```

5 Technical background

In the following, I will discussed the algorithms that are implemented in Jellyfish for the simulation of strong coupling patterns. This part is both a documentation, as well as a tutorial on how to approach these calculations. Any suggestions to improve the algorithm are appreciated.

5.1 Basics

In order to simulate a spectrum, we need to know which frequencies are present, and the amplitude of these frequencies. Based on that, we can draw a spectrum directly (as a histogram), or simulate the FID (as a series of complex exponentials). Calculating the spectrum directly is much faster, and this is used in Jellyfish. This is accurate if the number of points is high enough.

In order to calculate the energies, we need the Hamiltonian of the system. The total Hamiltonian is the Hamiltonian of all spins, including the Zeeman interaction (with the chemical shift) and the scalar coupling (J-coupling) interaction between all pairs of spins:

$$\mathscr{H} = \sum_i \mathscr{H}_i^{\text{zee}} + \sum_i \sum_{s>i} \mathscr{H}_{i,s}^{\text{J}} \tag{1}$$

Here, the double sum is only needed for a part of all pairs ([i,s] and [s,i] are equal, and [i,i] = 0).

To get the energies of the transitions, we need to transform the Hamiltonian to a diagonal frame (i.e. gets its eigenvalues)[1]:

$$\mathscr{H}^{\text{int}} = T^{-1} \bullet \mathscr{H} \bullet T \tag{2}$$

where the 'int' superscript indicates we have gone to the interaction frame. Here T represents the eigenfunction matrix, we we can use to transform the Hamiltonian to a new frame. T^{-1} is the inverse of this matrix. In this representation, the energy of transition $i \rightarrow j$ is equal to $\mathscr{H}^{\text{int}}[i,i] - \mathscr{H}^{\text{int}}[j,j]$.

Apart from the energy, we also need the transition probability, i.e. the intensity of the transient. This depends on two things: the start operator, and the detection operator. We use the following definitions[2]:

$$\rho_0 = I_x \tag{3}$$

as a starting operator. And

$$\text{Detect} = I_+ \tag{4}$$

To get the probability of a transition, we multiply these values in a point-wise fashion: $\rho_0 \cdot \text{Detect} = I_x \cdot I_+$. However, we want this result to be defined in the interaction frame:

$$P = (T^{-1} \bullet I_x \bullet T) \cdot (T^{-1} \bullet I_+ \bullet T) \tag{5}$$

Having calculated this, we can get the intensity of transition $i \rightarrow j$ as $P[i,j]$.

5.2 Hamiltonians

The Zeeman Hamiltonian is (including Chemical shift effects):

$$\mathscr{H}_{\text{zee}} = \sum_i B_0 \gamma_i (1 + \delta_i) I_z^i \tag{6}$$

with B_0 the magnetic field strength in Tesla, γ to gyromagnetic ratio of the nucleus of interest in Hz/T and δ the chemical shift.

The scalar coupling Hamiltonian (J-coupling) is:

$$\mathscr{H}^{\text{J}} = \sum_i \sum_{s>i} \mathscr{H}_{i,s}^{\text{J}} = \sum_i \sum_{s>i} J_{i,s}(I_x^i I_x^s + I_y^i I_y^s + I_z^i I_z^s) \tag{7}$$

[1]Note that I use the $\bullet$ symbol to represent a matrix multiplication, and a $\cdot$ for a point-wise multiplication.

[2]Note that I use I_x to mean the Pauli matrix of the $\hat{I}_x$ operator. I will do this throughout this manual. All operators are in their matrix form.

with $J_{i,s}$ the scalar coupling value between nucleus i and spin s.

Often, this Hamiltonian is split in apart that commutes with the Zeeman term (i.e. the weak coupling part) and a part that does not commute: the strong coupling:

$$\mathscr{H}^{\mathrm{J}} = \mathscr{H}^{\mathrm{J}}_{\mathrm{weak}} + \mathscr{H}^{\mathrm{J}}_{\mathrm{strong}} \tag{8}$$

with:

$$\mathscr{H}^{\mathrm{J}}_{\mathrm{weak}} = \sum_i \sum_{s>i} J_{i,s}(I^i_z I^j_z) \tag{9}$$

and

$$\mathscr{H}^{\mathrm{J}}_{\mathrm{strong}} = \sum_i \sum_{s>i} J_{i,s}(I^i_x I^s_x + I^i_y I^s_y) \quad . \tag{10}$$

Jellyfish makes use of the fact that the full Hamiltonian is always Hermitian: the lower diagonals are the complex conjugate of the upper diagonals. Therefore, we construct a matrix with only the upper diagonals, which still leads to the correct eigenfunctions an eigenvalues using the `numpy` `eigh` routine. To demonstrate why this is of use, I will describe some equalities using the `triu` function. This function returns the matrix with only the diagonal, and upper diagonals retained.

We can rewrite the equation for the strong J-coupling in a more convenient way:

$$\mathscr{H}^{\mathrm{J}}_{\mathrm{strong}} = \sum_i \sum_{s>i} J_{i,s}(I^i_+ I^s_- + I^i_- I^s_+)/2 \tag{11}$$

We can then write:

$$\mathrm{triu}(\mathscr{H}^{\mathrm{J}}_{\mathrm{strong}}) = \sum_i \sum_{s>i} J_{i,s}(\mathrm{triu}(I^i_+ I^s_-) + \mathrm{triu}(I^i_- I^s_+))/2 \tag{12}$$

$$= \sum_i \sum_{s>i} J_{i,s}(I^i_+ I^s_-)/2 \tag{13}$$

Here, we use the fact that $I^i_+ I^S_-$ only has non-zero values on the upper diagonals, and $I^i_- I^S_+$ has no non-zero values on the upper diagonals. Naturally, this is faster to calculate than the full Hamiltonian.

5.3 Operator definitions

To set up the Hamiltonian, as well as the start and detect operator, we need several one and two spin operators to be defined. Above, I used I to represent a spin operator. Note that, in the Hamiltonians described above, I must stand for the spin operator defined in the full basis (i.e. taking all the spins into account). Below, I shall use $\mathfrak{I}$ to represent a spin operator in its own basis.

In order to calculate, say, the I^1_z operator (i.e. the z-operator of spin 1 in the full basis), we take the operator in its own basis ($\mathfrak{I}^1_z$) and use the Kronecker product ($\otimes$) to come to the full basis. For a 3 spin system:

$$I_z^1 = \mathfrak{I}_z^1 \otimes \mathbb{1}^2 \otimes \mathbb{1}^3 \tag{14}$$

Here, $\mathbb{1}^i$ represents the identity matrix of spin i. The Kronecker product works like:

$$\mathfrak{I}_z^1 \otimes \mathbb{1}^2 = \begin{bmatrix} -1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -1/2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 0\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1/2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} \tag{15}$$

$$= \begin{bmatrix} -1/2 & 0 & 0 & 0 \\ 0 & -1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \end{bmatrix} \tag{16}$$

for two spin 1/2 nuclei. Note here that this calculation is quite inefficient: a lot of zeroes are calculated. Actually, for an I_z operator in matrix form, we know beforehand that we can expect only non-zero values along the diagonal.

In all cases, we only need the upper diagonal definitions of the operators. This is because we use the Hermitian property of the Hamiltonian, and we need only the upper diagonal transition probabilities. This means that we can use:

$$\mathrm{triu}(I_x) = I_+/2 \tag{17}$$

which means we only need I_+ (and not I_y, I_x or I_-). This is because I_- is the complex transpose of I_+, and is therefore easily related to it.

The ideas described above lead to the fact the we need to know Iz and I_+ for each spin. We should establish these as efficient as possible. For both of these, we can generate them as 1D arrays in first instance: both operators have only a single 'order' diagonal populated. For I_z Jellyfish uses a 1D Kronecker product operation.

5.4 Detection

For the detection we need both I_x (the start operator) and I_+ (the detection operator). Using Equation 17, we can write this more efficiently:

$$\mathrm{triu}(P) = (T^{-1} \bullet \sum_i I_x^i \bullet T) \cdot (T^{-1} \bullet \sum_i I_+^i \bullet T) = 1/2(T^{-1} \bullet \sum_i I_+^i \bullet T) \cdot (T^{-1} \bullet \sum_i I_+^i \bullet T) \quad . \tag{18}$$

In this way, the two parts are equal, and only need to be calculated once. Note that this is only valid if *all* spins are detected. In the case were we do not detect some spins, these should not be included in the I_+ part (the detection operator) but must be include in the I_x part. This makes the sums unequal.

5.5 Speeding up: block diagonals

One efficient way to speed up the calculation of the spectrum is to separate the total matrix representation of the system into different blocks. We can do this if there are eigenfunctions who have no matrix elements between them (i.e. no off-diagonal element). These off-diagonal terms are cause by the strong J-coupling elements. The challenge is to find these blocks in an efficient way.

The first trick we can use is called 'total spin factorization'. In here, we separate the Hamiltonian in parts which have equal total spin quantum number. This is based on the fact that our J-Hamiltonian has $I_+^i I_-^S$ as a term. In order for there to be a non-zero value between two states, the spin quantum number of spin s should be able to lower 1, and of spin i to be raised one. This leads to a new state with the same total spin quantum number! Therefore, two states that have a different total spin quantum number can never be related, and can be treated as separate groups, i.e. as blocks within the Hamiltonian.

The above is true in all cases. However, we know that if there are no J-couplings present, *all* states are blocks of there own. This means that there may be smaller blocks depending on the actual problem. In order to find them, we make a list of all connections between states. We then find all independent groups via the breadth first search (an algorithm from graph theory). This takes time, but is often worth it in the end.

All found blocks are diagonalised on their own. In the detection, the eigenvectors of two blocks are needed to find the intensities of the transitions between them (the detection operator *does* have terms between these groups!). This transformations via the T terms are also quicker in this way, then for the full Hamiltonian.

5.6 Speeding up: Composite Particle Model

Another way to speed up calculations concerns the case where we have multiple identical spins. In Jellyfish, this is input using the 'multiplicity' option. Multiple spins are identical if they have the same spin quantum number, chemical shift, and identical couplings (if any) to all other spins. A clear case where this is true is for the ^{1}H nuclei in a CH_3 group.

Normally, we would treat this group as a system with $2^3 = 8$ energy levels. However, we know that there can be no off-diagonal (J-terms) between these. Looking at this system in more detail, we see that there are four energies, with intensities [1,3,3,1]. The trick that we can use, is to spilt this system in multiple individual parts. This called the Composite Particle Model (CPM). In this particular case, we can split this system of three spin 1/2 nuclei into one spin 3/2 nucleus (intensities [1,1,1,1]) en two spin 1/2 nuclei (intensity [0,2,2,0]). This gives us a size 4 and size 2 problem, instead of a single size 8 problem. This is much faster, especially if the rest of the spin system is quite large. Note that the two spin 1/2 subsystems are identical, so we calculate its spectrum only once, and multiply by 2.

The method described above is valid for any group of identical nuclei. The intensities follow exactly the same pattern this group of spins cause as a J-coupling effect on other spins. For the example above, the three ^{1}H spins cause a quartet ([1,3,3,1]). For four spins, a quintet would ensue ([1,4,6,4,1]) which can be described as a spin 2 system, three times a spin 1 system, and two times a spin 0 system.

Additional speedup can be achieved using the CPM by using symmetries. We use CPM to split all the identical parts of the full spin system. We then have a whole set of individual subsystems that we need to do the spectrum calculation on. Before we do that, we can check if there are copies due to symmetry effects. Doing this doe snot require a user input for the symmetry, but can just be check for all cases (naturally, user input to supply the symmetry will also work, but Jellyfish is written to be generally applicable).

6 Contact

To contact the `Jellyfish` team write to `ssnake@science.ru.nl`.