The xpy3-tools tutorial

Oleg V. Petrov | KFNT MFF | Univerzita Karlova

Version 1.2

30th January 2025

Table of Contents

1 Introduction

`xpy3-tools` is a collection of Python scripts for NMR data processing intended to be used as command line tools in Bruker Topspin software. As such they rely on the Bruker Python API available in Topspin 4.1+ [1]. The scripts primarily target the author's needs and the scope of his own practice, but might be of interest to other NMR practitioners too. The present version (v1.2) comprises 13 scripts (Table 1), which number is supposed to be extended in newer versions, as time goes on and the tasks tend to grow.

The scripts are run by typing the command `xpy3 <script name>` in the Topspin command line. In this respect, they are similar to conventional Jython scripts or Bruker AU programs. Sharing a workspace with Topspin means, among other things, an access to the Topspin's GUI. Nonetheless most of `xpy3-tools` have their own GUIs popping up in a separate window, which is intended for interactive parameter optimization and visual control of the result before returning to Topspin. The GUI part is implemented using `matplotlib` / `Tkinter` methods.

Input / output (I/O) functionality is provided by the Python module `BrukerIO` from the package `JTutils` (https://github.com/jtrebosc/JTutils), for it is more advanced than the I/O functionality provided by the Bruker Python API as yet. None of `xpy3-tools` will modify in place `fid`, `ser`, `acqu(123)` and other files storing raw data (time-domain signals). If it is the time signal that is being processed, the data set will copied as a whole under a new `EXPNO` to keep the original time signal intact. Otherwise the data are processed in-place, the files of processed data being saved in `pdata` under the same `EXPNO` from whence the script has been run.

2 List of scripts

Script name	*Function*	*Dependencies**	*Comments*
`baseline`	Baseline correction in 1D	pybaselines [2] vam.whittaker [3]	Uses rolling standard deviation distribution method to identify baseline points [4] and Whittaker method for their interpolation [5].
`baseline2`	`baseline` applied to individual spectra in a pseudo-2D data set.	pybaselines [2] vam.whittaker [3]	Enables navigation across 2D dataset for baseline adjustment of individual spectra.
`cpmg_fp`	Whole-echo processing for CPMG echo trains.	pybaselines [2]	Relies on zero points in a CPMG signal when splitting it into echoes. To have those, consider using a `op_qcpmg` pulse program.
`cpmg_recursive`	T_2 relaxation time measurement on CPMG data with mono-exponential recursive model.	pybaselines [2]	Quantifies individual echoes, plots their cumulative sum vs time and fits it with a recursive model [6].
`cpmg_relaxation`	T_2 relaxation time measurement on CPMG data using mono- or stretched-exponential model.	pybaselines [2]	Plot individual echo intensity vs echo time and fits the curve with mono- or stretched exponential.
`denoise`	Low-rank SVD approximation of 1D signals in time domain.		Uses a Toepitz matrix format in a 2D representation of FID; the SVD thresholding used is due to Gavish and Donoho [7].
`denoise2`	Approximation of 2D signals in time domain using noise-adapted principal component analysis		Runs NUS on the acquired data to randomize principal components and thereby to average out their noise-related content before using them in data reconstruction [8].
`icoshift`	Peak alignment over a series of spectra stacked in a pseudo-2D dataset or a multi-display mode.	pybaselines [2] pyicoshift [9]	A number of peak regions can be selected for individual alignment, or you can rely on one peak to shift the whole spectrum.
`phase_fid`	Zero-order phasing of 1D signals in time domain.		A handy tool to maximize the Re part of an FID or echo train, *e.g.* for quantitation purposes.
`rotsync`	Reconstruction of 1D rotor-syncronized MAS spectra from arbitrarily sampled FIDs.		Useful in quantitation of different chemical shift components through direct integration; facilitates line shape analysis.
`rotsync_plus`	Includes a more involved echo quantitation than `rotsync`.	vam.whittaker [3]	In theory should provides better SNR than `rotsync`.

`symmetrize`	MAS sideband symmetrization for 1D quadrupolar spectra of half-integer spins.		Facilitates fitting of first-order quadrupolar model to MAS spinning sidebands; good for presentation.
`t1t2`	Basic relaxation analysis	pybaselines [2] vam.whittaker [3]	Interactive integration of pseudo 2D spectra with exponential fit.
`utils`	Contains common utility functions and classes.	contourpy [10]	Ever growing collection of `def`s.

*) Specific for a given script; for common dependencies see `requirements.txt`.

3 Installation

3.1 Python environment setup

Topspin 4.2+ comes with a preinstalled Python 3 environment in `<topspin>/python/lib/python3.x/`, which includes Bruker API and a few packages used in example scripts. To use a Python environment running outside Topspin, you should specify the path to the chosen Python 3+ interpreter on the Topspin preferences page as `Preferences` → `Python 3+` → `Select Python 3+ Environment`.
Imagine you have chosen to use a Python 3.x interpreter installed, *e.g.*, in `/usr/local/bin` on your Linux machine. Imagine also that the package installer for Python is `pip`, namely `pip3.x`. In the Linux command line, navigate to the Python interpreter folder, *e.g.* to `/usr/local/bin`, then run the command

```
python3.x pip3.x install -r path/to/requirements.txt
path/to/ts_remote_api*.whl path/to/bruker_nmr_api*.whl
```

to install the required Python packages. The configuration file `requirements.txt` is taken from the `xpy3-tools` project directory. The `.whl` files are taken from `<topspin>/python/examples` or, should you use Topspin 4.1, you can download these files from bruker.com. The procedure should work on a Windows machine too (not tested though).

A special care must be taken of the module `BrukerIO`. The package `JTutils` containing this module cannot be installed via `pip`, hence not included in `requirements.txt`. To work it out, copy the file `brukerIO.py` directly from the `JTutils` at `https://github.com/jtrebosc/JTutils/blob/master/CpyLib/` homepage and place it in the folder where `xpy3-tools` scripts are kept (see the next paragraph).

3.2 Placement of scripts

The syntax `xpy3 <script-name>` to run `xpy3-tools` scripts from the Topspin command line implies a proper script location. A search path for Python 3 scripts is hard-coded to `<topspin>/python/examples`. Therefore, the simplest way to make it work is to place `xpy3-tools` scripts in that folder. Should you prefer another location, add a path. In a text editor, open the file `<topspin>/exp/stan/nmr/py/xpy3.py` (you might need administrator's privileges for that), then find and comment out the line

```
return getParfileDirs(PYTHON_3_INDEX)
```

Then add the following code below that line (mind indents):

```
    defaultDir = getParfileDirs(PYTHON_3_INDEX)
    extras = ['/my/path/to/scripts']
    for x in extras: defaultDir.append(x)
return defaultDir
```

You may add as many paths in `extras` as needed, separating them by a comma. In fact, any `xpy3-tools` script can be run from a console command line as a usual Python program, as `python3.x <script name>`, or be launched from within a dedicated IDE such as Spyder or VSCode. All you need is a target NMR dataset being opened in a Topspin window and the embedded web service running [1].

4 Usage examples

4.1 baseline

This script implements interactive baseline correction in 1D. For the baseline classification part, it employs a function `std_distribution()` from the package `pybaselines` [2]. The function identifies baseline points by analyzing a rolling standard deviation distribution across a spectrum. It is controlled by two parameters – `half_window` and `num_std` – to adjust, respectively, the number of points involved in the rolling standard deviation calculation and the number of standard deviations for baseline points thresholding. For the baseline interpolation part it employs a Whittaker smoother `ws2d()` from the package `vam.whittaker` [3]. This adds the third adjustable parameter – `lambda` – to control the interpolation smoothness. In the Topspin command line, you run `xpy3 baseline` (Fig. 1) which brings about a two-graph window (Fig. 2). The upper graph shows the classified baseline points and baseline interpolation overlapping the original spectrum. The lower graph shows the spectrum after subtracting the baseline thus estimated.

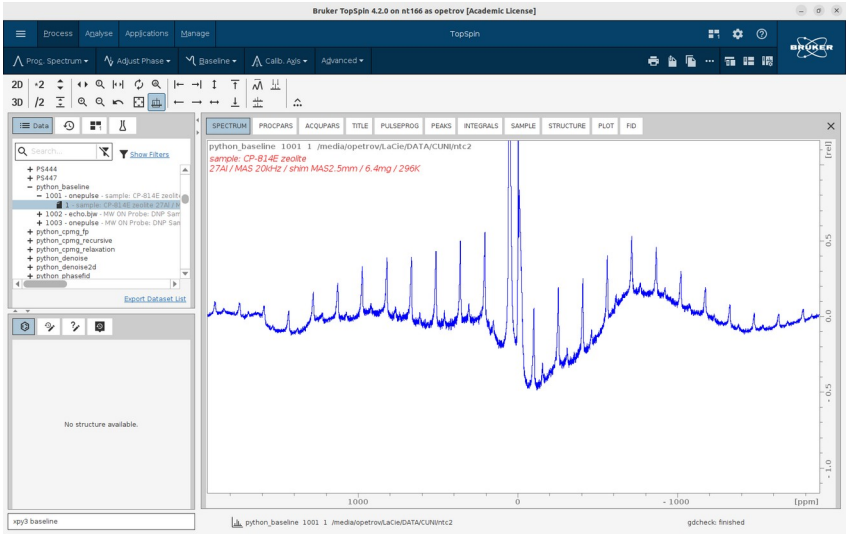

Fig. 1

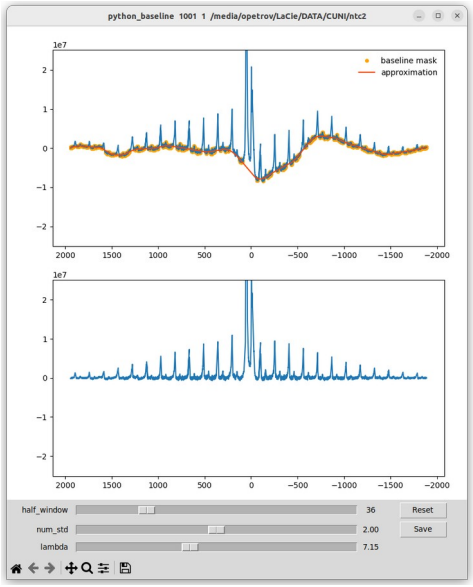

Fig. 2

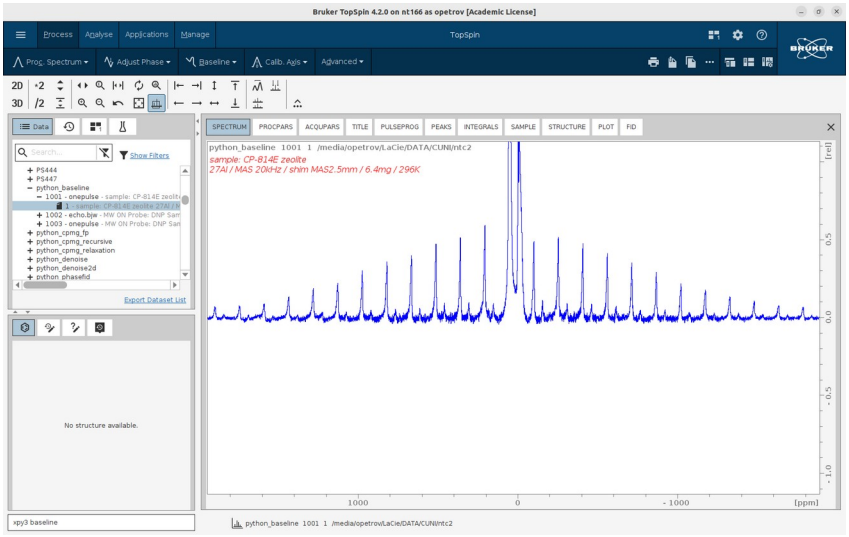

Fig. 3

You can try and improve the baseline estimate manually using sliders. Clicking a `Reset` button will restore the automatic estimate. Clicking `Save` writes the spectrum shown on the bottom graph to the `pdata` folder and updates the Topspin window accordingly (Fig. 3). To exit without updating, close the GUI with ☒ or `Alt+F4`.

The baseline correction is applied to both Re and Im parts of the spectrum, provided that `1r` and `1i` file modification times coincide. It might be useful if you want to process the spectrum any further using both these parts (phase adjustment, conversion to magnitude mode, inverse FT, etc.). The script is also applicable to data sets with `1i` file missing, *e.g.*, to magnitude spectra.

4.2 baseline2

Here, the baseline correction is applied to a series of spectra stored in pseudo-2D datasets, that is, in `2rr` and `2ii` files (but not in `2ir` or `2ri` files). For instance, Fig. 4 shows a series of 24 spectra from a T_1 inversion-recovery experiment on methane compressed inside MOF powder.

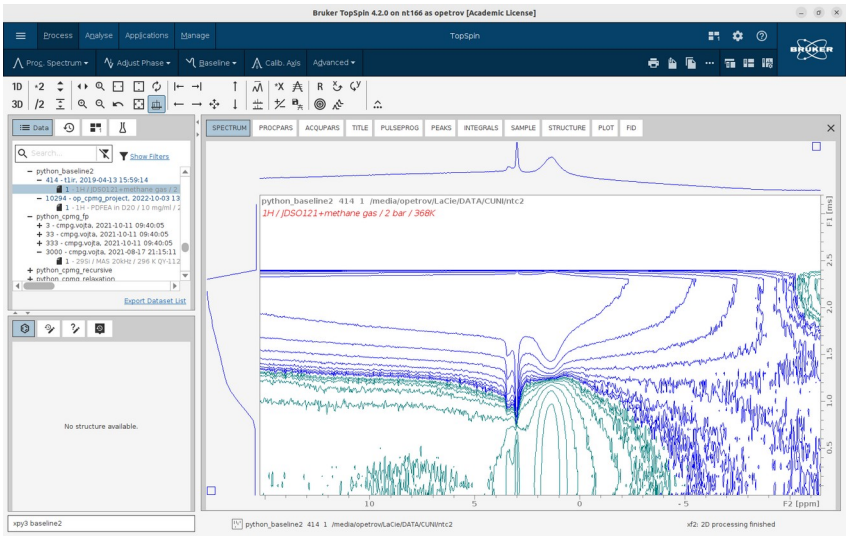

Fig. 4

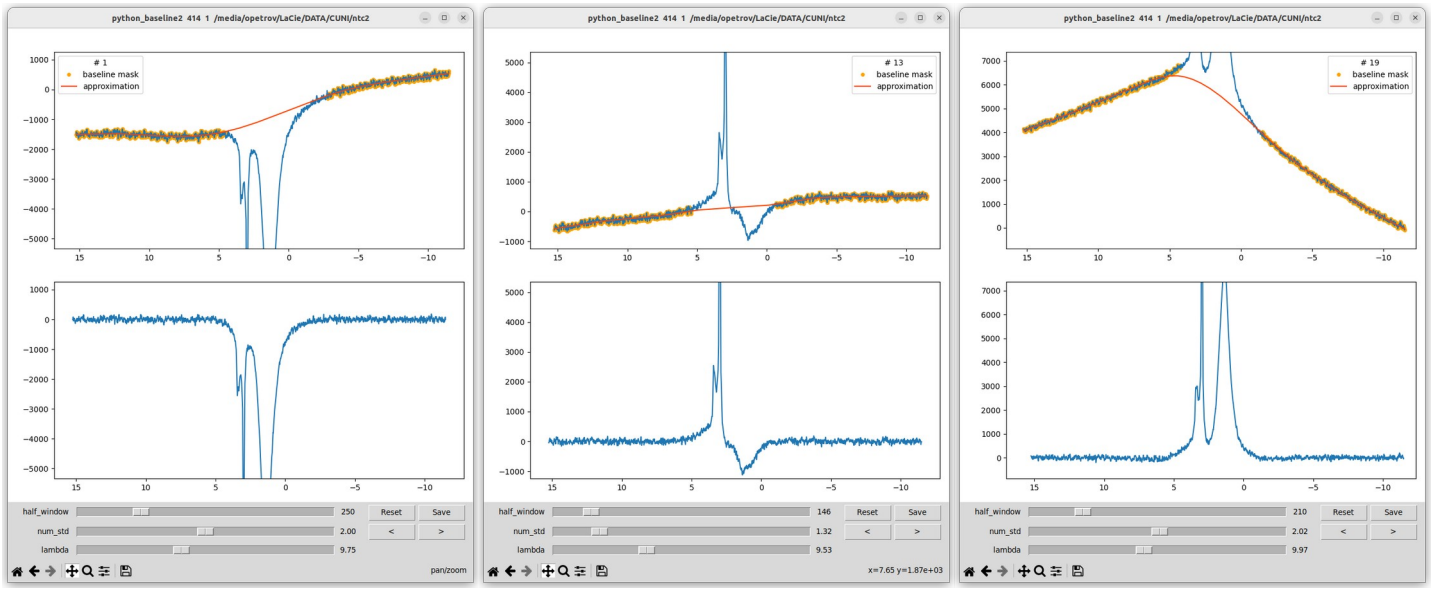

Fig. 5

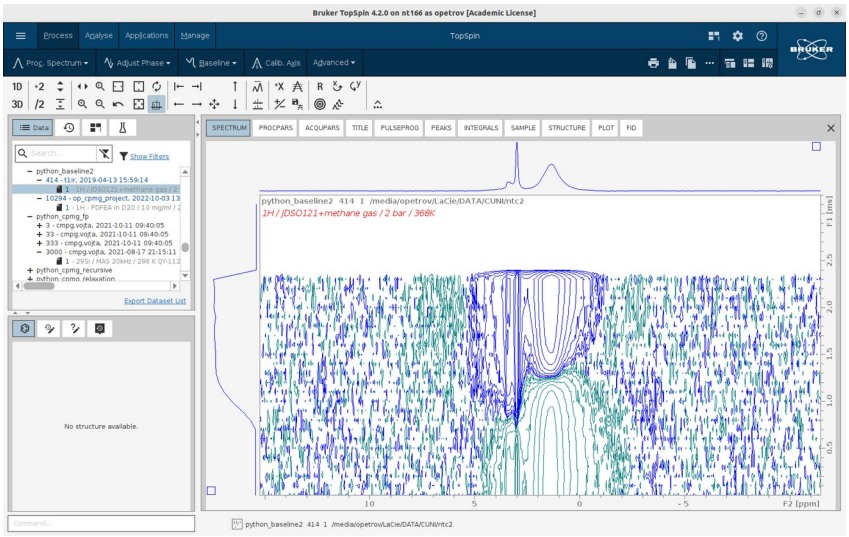

Fig. 6

Running the `xpy3 baseline2` command brings about a pop-up window identical to Fig. 2 but with the option to navigate across spectra using the buttons ">" and "<". By default, the first spectrum of a series is displayed. Once you inspect and optimize, if needed, the baseline estimate, you switch to another spectrum, and so on (Fig. 5). Eventually you click `Save` to write the entire series to the disk (Fig. 6).

Tip: If you find matplotlib too slow at re-drawing spectra in interactive mode (at zooming and such), you can try and improve it by increasing a line segment simplification parameter `path.simplify_threshold` in the matplotlib configuration file `[PYTHON]/.../matplotlib/mpl-data/matplotlibrc` (upper limit is 1).

4.3 cpmg_fp

This script is intended to be used with `qcpmg` type pulse programs which generate continuously sampled echo decays (aka echo trains). When processing an echo decay as a regular fid – *e.g.* with the command `fp` (FT + phase correction) – it results in a discrete 'spikelet' spectrum. To get a regular continuous spectrum, you have first to collapse the echo train into a single echo (sum of echoes) and follow the whole-echo processing steps, *i.e.*, split the echo in the middle, swap the halves and FT. This is what `cpmg_fp` does, plus it phases the spectrum to maximize the Re part.

Fig. 7 shows a train of 256 spin echoes from a zeolite sample (^{29}Si NMR). Typing `xpy3 cpmg_fp` brings about a two-graph window (Fig. 8) with the top graph showing the echo decay signal (same as in a Topspin window), while the bottom graph shows the average echo (sum of echoes) after first splitting the signal into individual echo segments. By default, all echo segments are included in the sum, but you can change this with the slider (`# echoes`). Changing `# echoes` affects both SNR and the lineshape of the spectrum provided it comprises differently relaxing components (different T_2's). Once you are certain about `# echoes`, click `OK` to get the spectrum (Fig. 9).

For the script to work, the echoes in the train must be separated by one or more zero data points. A pulse program `op_qcpmg` attached to the package is a special modification of a standard Topspin `qcpmg` program to include such zeroes in acquisition.

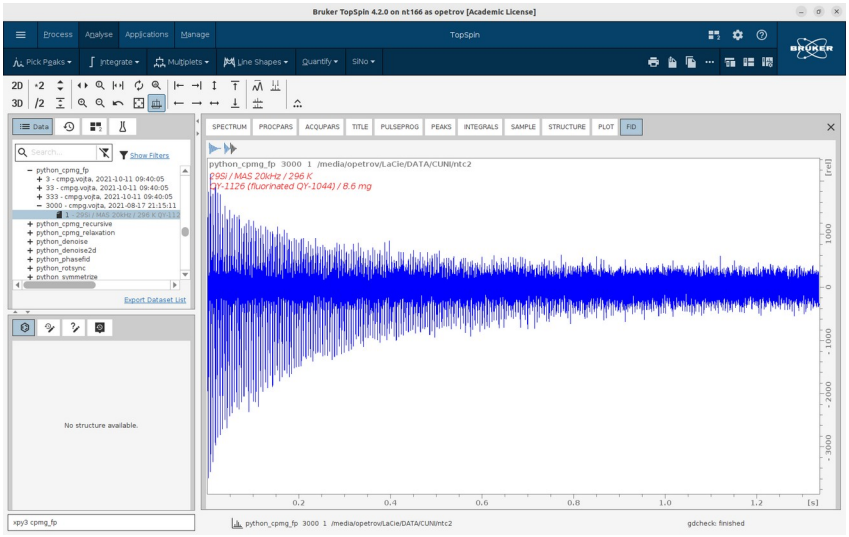

Fig. 7

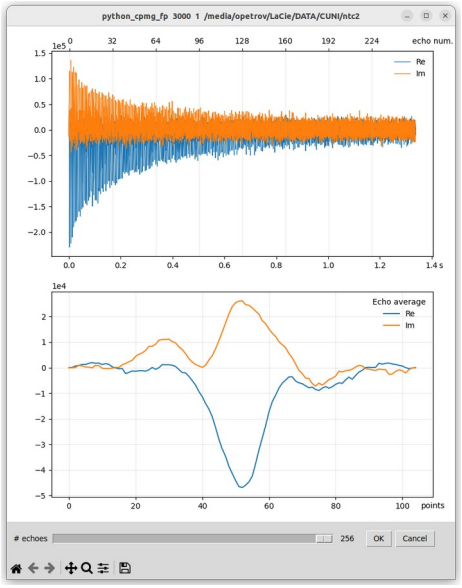

Fig. 8

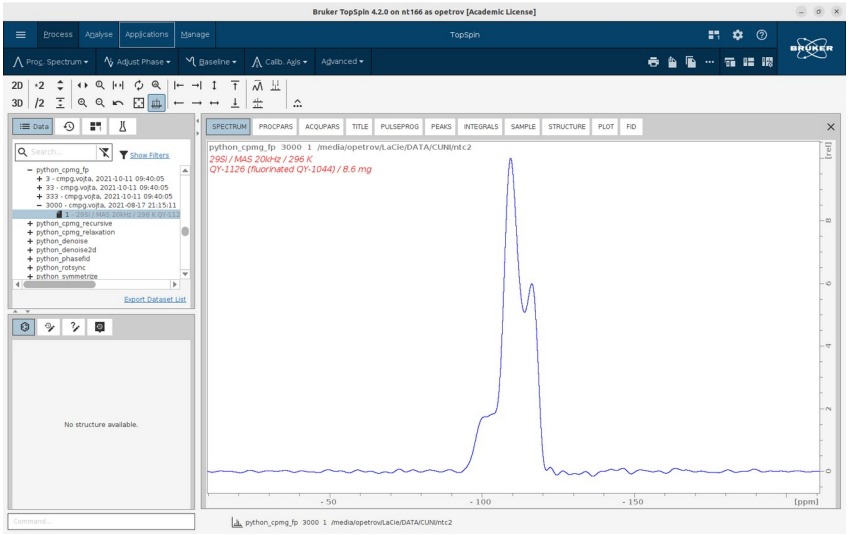

Fig. 9

4.4 cmpg_relaxation

This script again must be used in combination with `qcpmg` type pulse programs that generate continuous trains of zero-separated echoes. Given the zero positions, the script splits the train into individual echoes and quantifies them using either the midpoint intensity or integral intensity. The resultant echo decay curve – echo signal *vs.* echo time – can be fitted with either mono or stretched exponential model.

Fig. 10 shows an example echo decay for a fluorine-containing zeolite (^{19}F NMR) acquired with the pulse program `op_qcpmg` (included to the package). Running `xpy3 cpmg_relaxation` brings about a two-graph window (Fig. 11). The top graph shows the experimental signal, with detected echo positions marked with asterisks, and the bottom graph shows the echo decay curve, on a semi-log scale, measured by default from echo midpoints (with the option to switch to echo integrals). Clicking `Fit Me!` fits a mono-exponential model to the curve, the best-fit parameters being annotated (Fig. 12). In case of pronounced non-exponential decay,

there is an option to use a stretched exponential model with a mean relaxation time reported (Fig. 13). Clicking `Save&Quit` saves the decay curve in a two-column text file in `<EXPNO>/pdata/1/ct1t2.txt`, for further processing (*e.g.* for Inverse Laplace transformation). Note that the annotation text with best-fit model parameters will not be saved, so mind that you write them down before clicking `Save&Quit`.

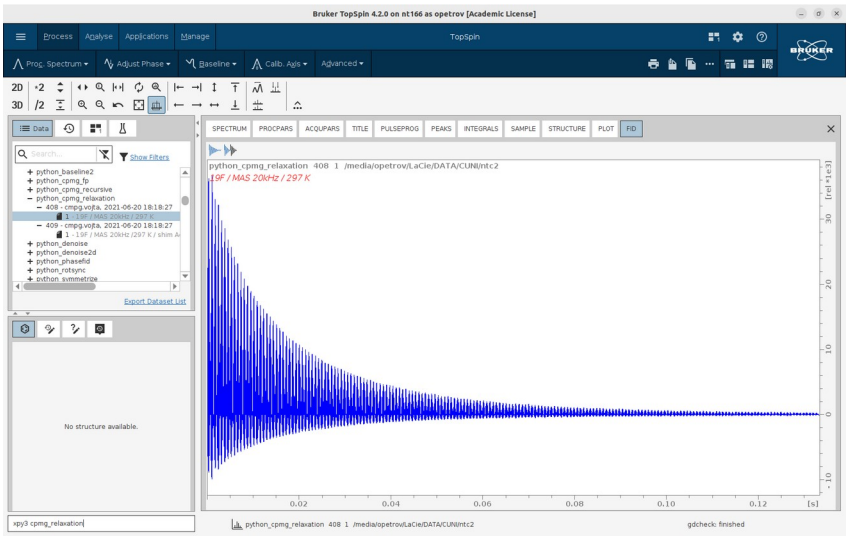

Fig. 10

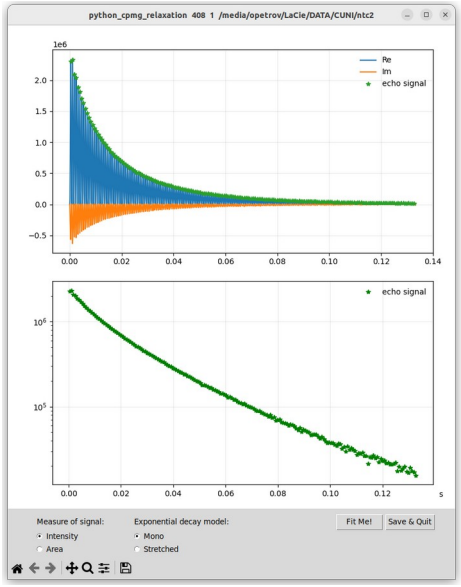

Fig. 11

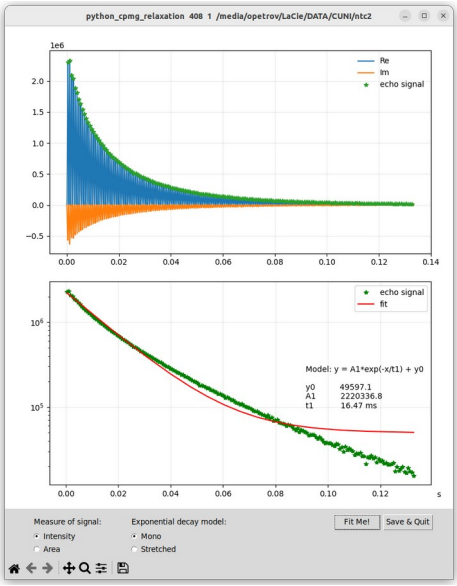

Fig. 12

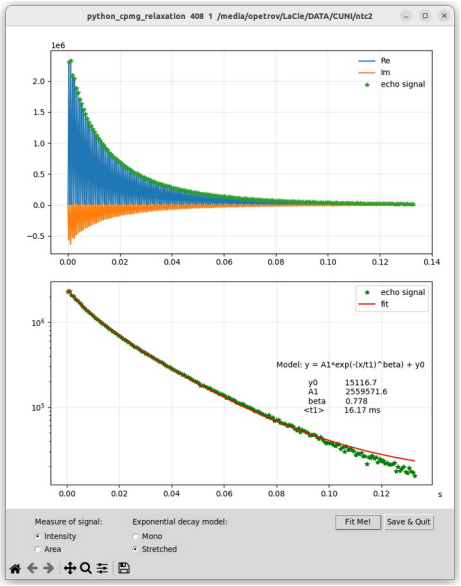

Fig. 13

4.5 cpmg_recursive

The functionality of this script is identical to `cpmg_relaxation` except for its using a cumulative sum of consecutive echoes as the measure of relaxation instead of individual echo intensity. The cumulative sum of echoes is fitted with a recursive mono-exponential relaxation model [6]. The advantage of this approach is two-fold: (i) the summation decreases a scatter in points on longer relaxation times; (ii) the best-fit parameters are less sensitive to the initial, fast-relaxing data points and thus better capture the overall relaxation trend. Fig. 14

shows the performance of `cpmg_recursive` on the same data as in Fig. 10. The GUI controls are analogous to `cpmg_relaxation`.

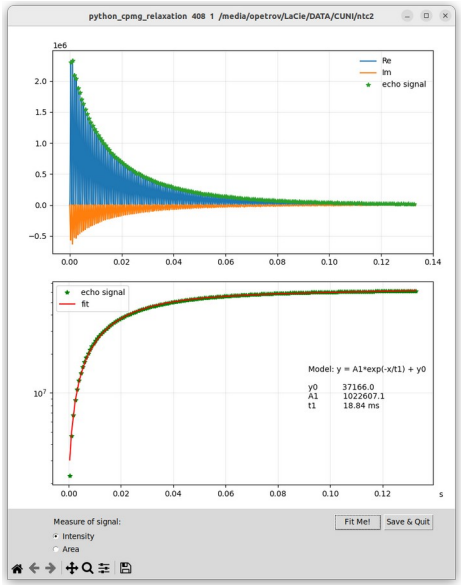

Fig. 14

4.6 denoise

This script performs a Cadzow-like noise removal from 1D time-domain signals. The Cadzow's method [11] is based on a low-rank decomposition of a data matrix, hence at the first step the signal has to be converted into a matrix where each row is a shifted version of the signal (aka a Toeplitz matrix). Next, the matrix is subjected to a singular value decomposition (SVD) and only some largest singular values are retained. Then, the retained terms are employed to reconstruct (approximate) the matrix, which is finally converted back to a 1D signal. To avoid dealing with group delay's points preceding the actual data, the script automatically converts the signal to an AMX type ('analog filter' type) before denoising, via the Topspin command `convdta`.

The script does not include a GUI for interactive parameter optimization. For better performance, it is recommended to remove as much of the noisy tail in the FID signal as possible by setting the Topspin's proc. parameter `TDeff` somewhere between `TD` and the value at which truncation artifacts ('sinc wiggles') become visible. Another parameter the user can play with is the number of the largest singular values involved in the data matrix approximation. This number is set automatically using a Gavish-Donoho's (G.-D.) thresholding rule [7] but the user can adjust it running the script with an optional argument `offset`,

```
xpy3 denoise --offset n
```

where `n` is an integer (either positive or negative) which will be added to the G.-D.'s threshold value.

Fig. 15 shows a ^{13}C spectrum of an organic precursor inside a zeolite sample (on the top) and the same spectrum after applying `denoise` (on the bottom), demonstrating a flawless performance. Another example is a ^{13}C spectrum of alginate (Fig. 16), in which case `denoise` leaves two noisy spikes at 47 and 170 ppm. As a rule, such spikes appear 'off-phase' wrt real peaks and hence are readily recognizable as noise-related artifacts.

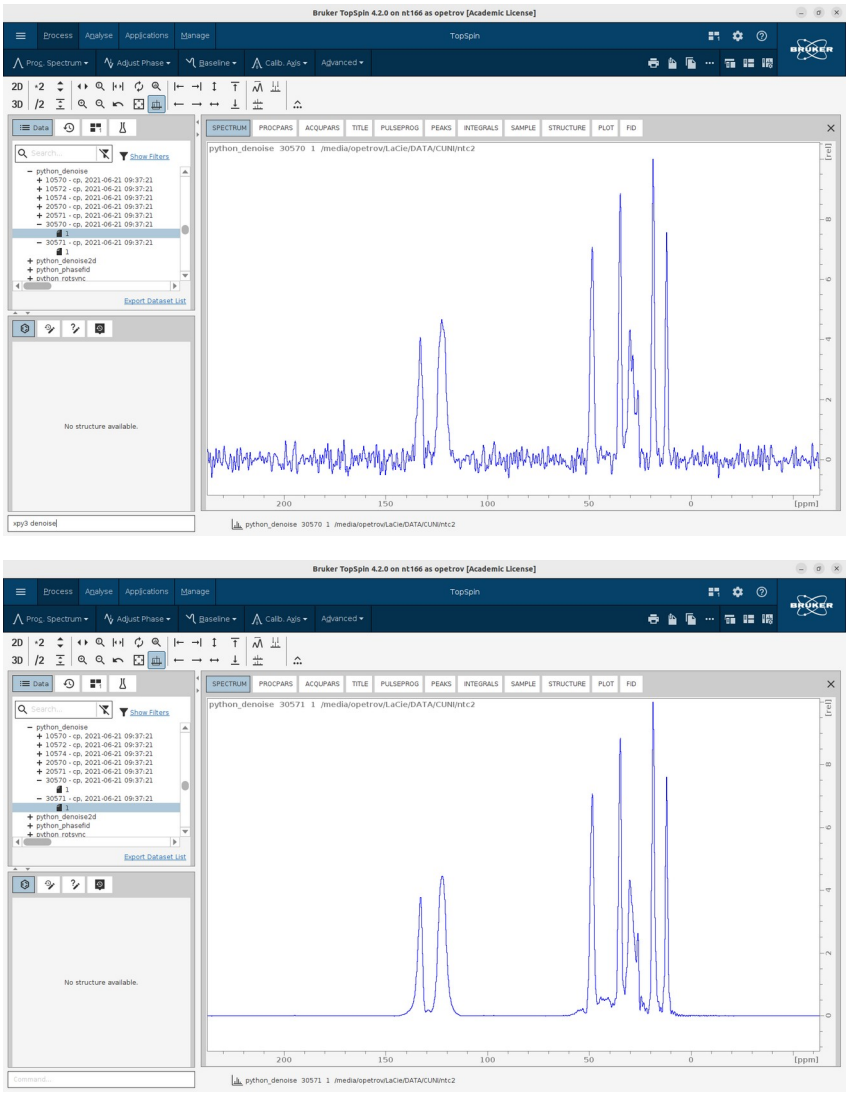
Bruker TopSpin 4.2.0 on nt166 as opetrov [Academic License]
Process
Analyse
Applications
Manage
TopSpin
Proc. Spectrum
Adjust Phase
Baseline
Calib. Axis
Advanced
SPECTRUM
PROCPARS
ACQUPARS
TITLE
PULSEPROG
PEAKS
INTEGRALS
SAMPLE
STRUCTURE
PLOT
FID
python_denoise 30570 1 /media/opetrov/LaCie/DATA/CUNI/ntc2
No structure available.
[ppm]
Bruker TopSpin 4.2.0 on nt166 as opetrov [Academic License]
python_denoise 30571 1 /media/opetrov/LaCie/DATA/CUNI/ntc2
No structure available.
[ppm]

Fig. 15

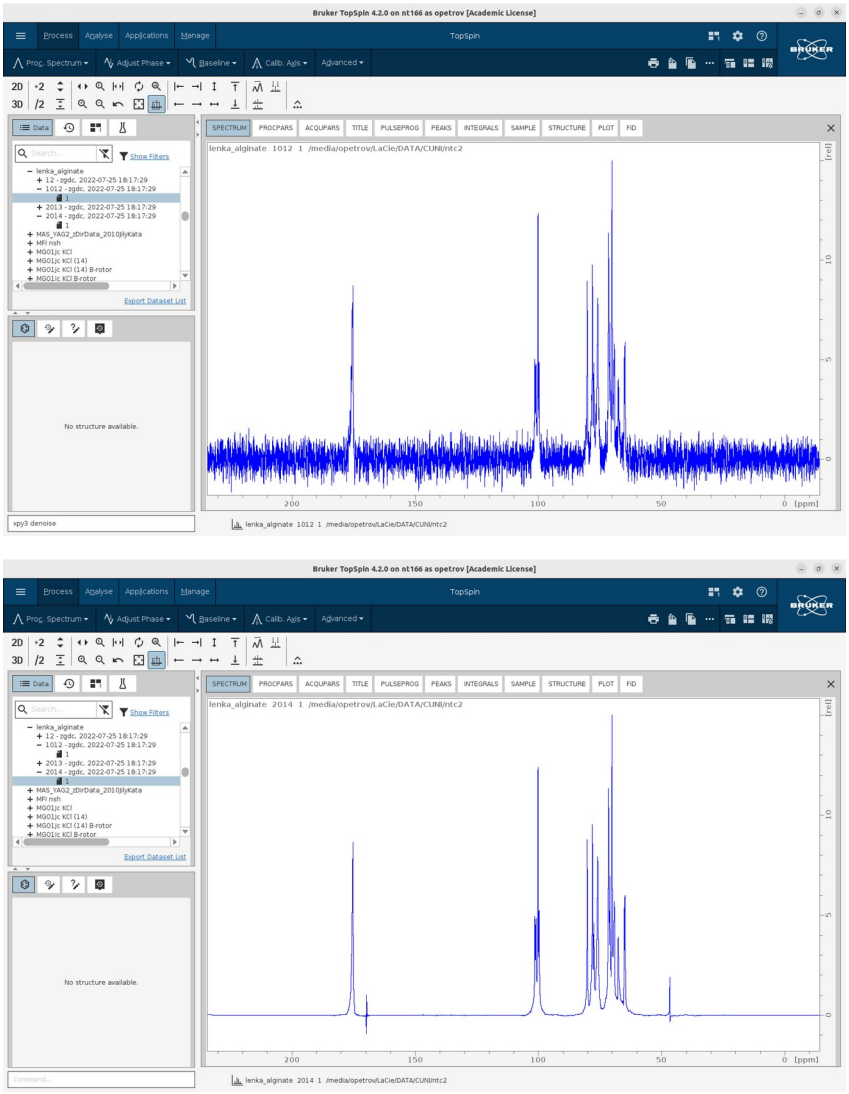

Fig. 16

4.7 denoise2

This script implements noise canceling in 2D by an original method [8]. Briefly, it is a variant of the PCA (SVD) based denoising where the principal components (PCs) constituting a basis for signal approximation are attenuated proportionally to their noise-related content. You may think of it as a regularized SVD approximation, as opposed to the conventional low-rank SVD approximation used in the 1D counterpart `denoise`. The PC attenuation is achieved through 'averaging' same-rank PCs in repetitive SVD on randomly chosen data samples. The size of the samples is controlled by a parameter `nus` in the range of 0.25 to 1.0 and the number of samples by the parameter `samples`, either or both these parameters can be passed through external arguments (the square brackets indicate their being optional):

```
xpy3 denoise2 [--nus nus] [--samples samples]
```

If the arguments are not passed, the default values `nus=auto` and `samples=20` are used.

Fig. 17 shows a phase-sensitive 2D INADEQUATE spectrum of strychnine in $CDCl_3$ (from Bruker's example dataset `/examdata/exam_CMCse_3/`). Beneath it is the same spectrum but after applying `denoise2`, demonstrating a factor of 2.3 improvement in SNR.

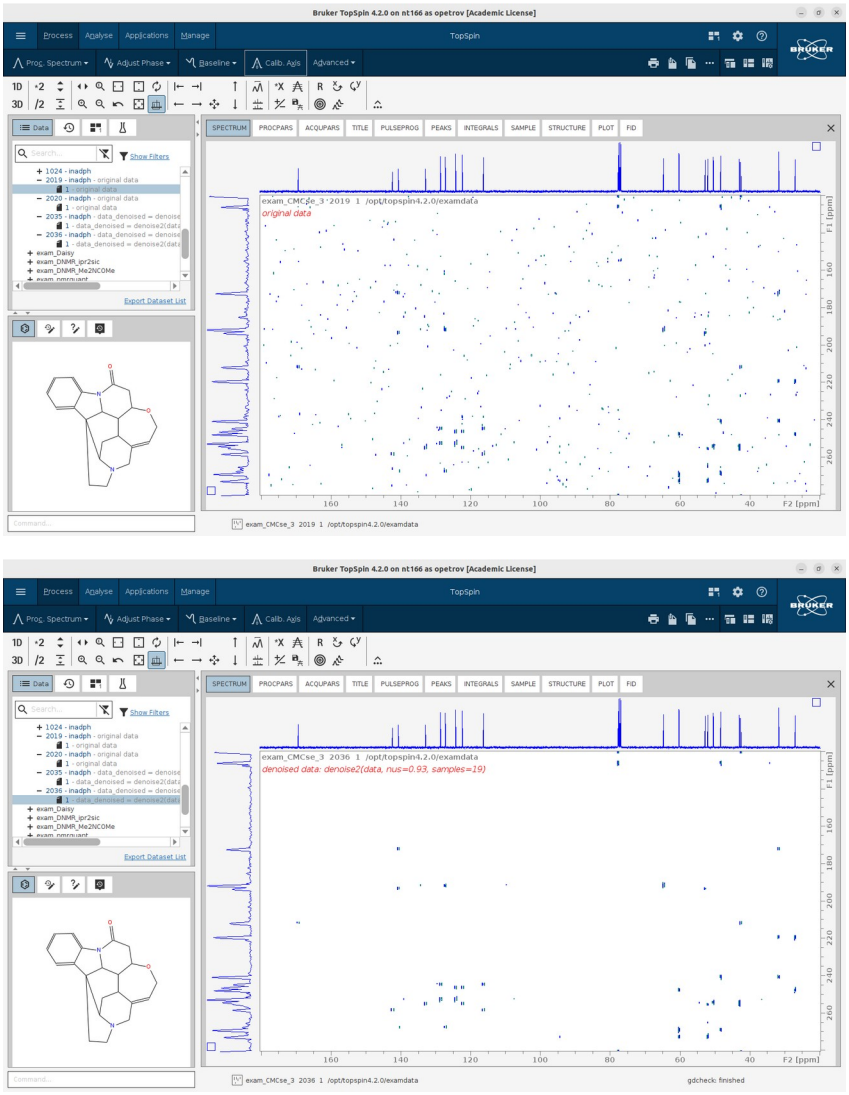

Fig. 17

Running `denoise2` on large datasets and with the number of samples greater than the default 20 may be painfully long (longer than a minute). Should you not want to wait, you could interrupt the script by closing an accompanying progress bar widget and restart it with a less number of samples. Like its 1D counterpart, `denoise2` copies the whole data set to a nearest available `EXPNO`.

4.8 icoshift

This is a frontend to utility functions of `pyicoshift` [9], a Python implementation of the *icoshift*. The name *icoshift* stands for *interval Correlation Optimized shifting* algorithm for peak alignment. It applies to spectra of same kind whose peak positions are prone to shifting because of fluctuations in pH, ion content, temperature

etc., or due to instrument factors (*e.g.* a drifting field). The `icoshift` is run either on a pseudo-2D dataset or on spectra from 1D datasets associated through a multi-display (`md`) mode, given the same size.

Fig. 18 shows spectra of a polyacrylamide hydrogel in D_2O from a variable-temperature experiment, opened in a Topspin's multi-display window (`.md`). Typing `xpy3 icoshift` brings about a two-graph GUI (Fig. 19). The top graph replicates the content of the Topspin's window, thus enabling peak intervals selection. Omitting the selection means treating the whole spectrum as one interval. Note that once selected, the intervals cannot be edited, you can only clear them altogether with a `Clear` button and start over. In Figs. 20 and 21, one interval is chosen that encloses a single peak from a reference compound (TMMS). With `Align mode` options, you can choose whether to align only this selected peak, keeping the rest intact (the option `n_intervals`), or to apply the shifts which will be found for this peak to the whole spectra (the option `whole`). When `n_intervals` is chosen, the gaps in the shifted spectra are filled with adjacent values, whereas for the `whole` option, the spectra are completed with `nan`'s on either of their sides. With the radio-buttons `Target`, you choose a reference signal to which the spectra are shifted to match its position. The default target is `maxcorr` ("*the signal with the highest correlation with all input signals*" [9]), the others are `average` (arithmetic mean of spectra), `median` (median of spectra), `max` (spectrum of maximum intensity), and `average2` (uses the average target spectrum twice). For more information about the targets and which one it is better to use, see a tutorial to the Matlab version of `icoshift` [12]. In practice, you just go through all the targets and watch the difference it makes.

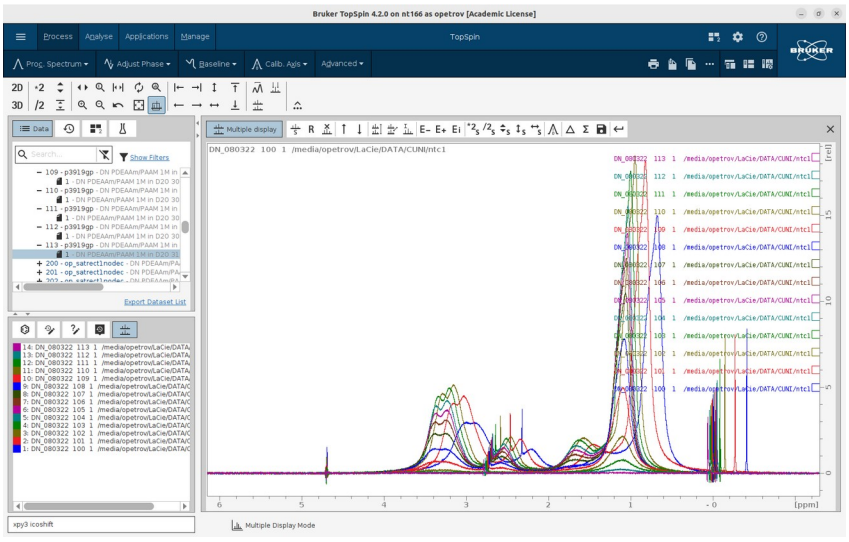

Fig. 18

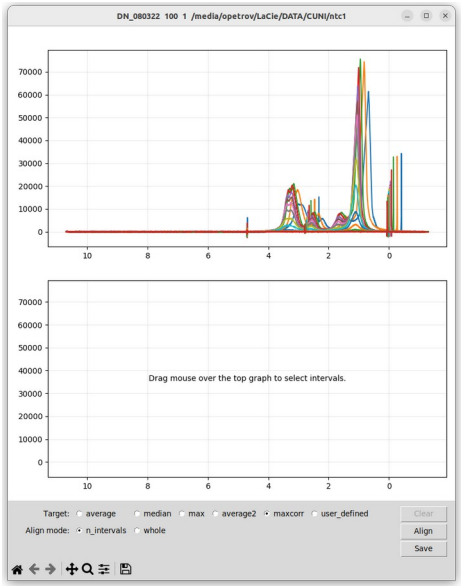

Fig. 19

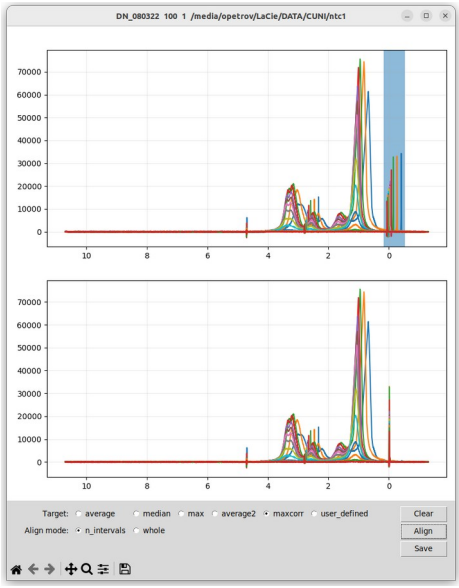

Fig. 20

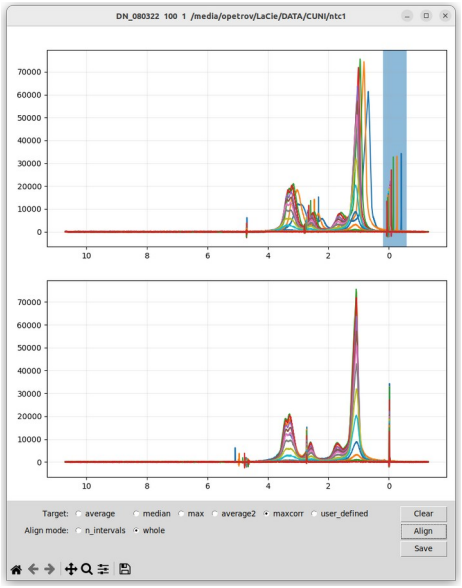

Fig. 21

4.9 phase_fid

This script implements zero-order phasing (rotation) of a signal in time domain, the first data point being used as a pivot. It is primarily intended for relaxometry where such a rotation is often needed to maximize the real part of a signal, for quantitation purposes.

Figs. 23 shows rotation of a CPMG echo train from Fig. 22 in order to maximize the real part of the echoes. The `phase_fid` saves the rotation under a new `EXPNO` , as it should when raw data are modified (Fig. 24). There is an option `Zero Im` to nullify the imaginary part of the rotated signal, which is a convenient and easy way to obtain a symmetric Fourier transform of the signal (the approach often employed in ^{2}H lineshape analysis). The value of the `PH_ref` slider can be written down and used to correct the homonymous Topspin's parameter for subsequent experiments.

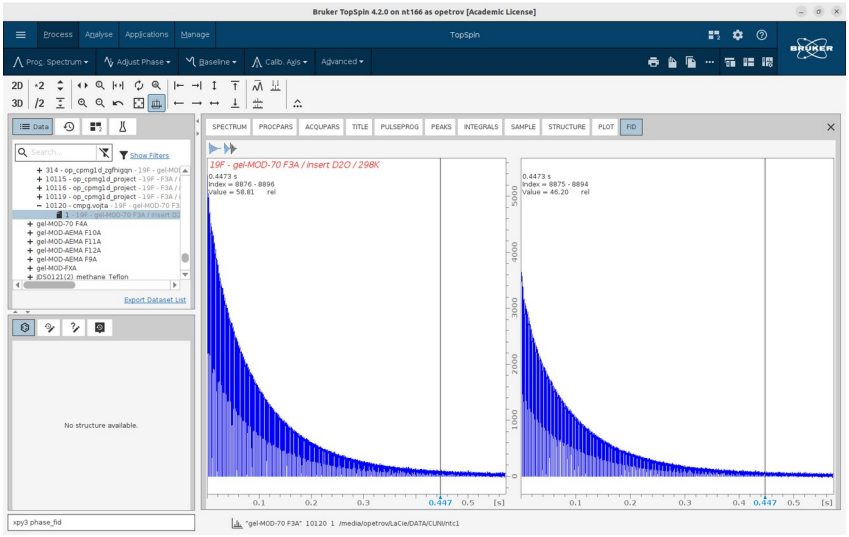

Fig. 22

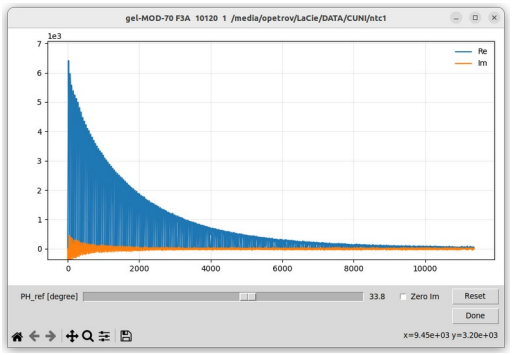

Fig. 23

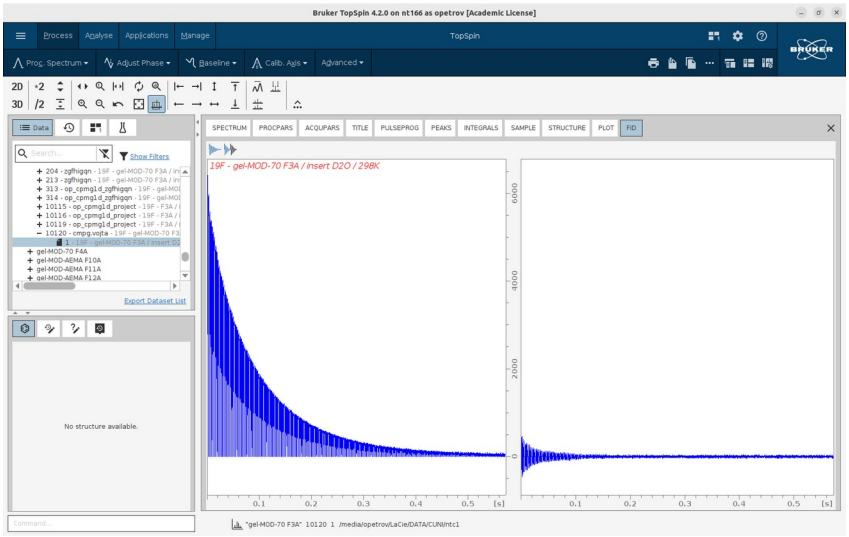

Fig. 24

4.10 rotsync

This tool is specific to magic angle spinning (MAS) NMR. What it does is, it positions rotary echoes in a 1D MAS signal and re-samples the signal down to these positions, thus rendering it a ‘rotor synchronized’ signal. The rotor-synchronized signal is one without spinning sidebands (ssb) in the spectrum, with satellite transitions intensity all falling into the centerband where it superimposes the central transition peaks. Thus, when it comes to quantitation of spectral components with different chemical shifts, it can be done directly through integration over the centerband. The lineshape of the rotor-synchronized spectrum is equivalent to the case of the infinite MAS rate, which is convenient for the line shape analysis. Usually, rotor synchronization is executed on the fly during signal acquisition. This, however, requires thorough adjustment of both sampling period and sampling offset as acquisition parameters. Since it is tedious and difficult to control, one might prefer this post-acquisition synchronization instead. Of course, it would require for the rotary echoes be well defined (detectable) over a few rotor periods.

Fig. 25 shows the centerband of a MAS spectrum of $Al(acac)_3$ acquired under a 20 kHz MAS (^{27}Al NMR). Running `xpy3 rotsync` brings about a window (Fig. 26) showing the given FID signal, in the magnitude mode, with rotary echo positions marked with asterisks. If needed, you can adjust these positions using the sliders `offset` and `period`. Clicking `OK` saves the echo positions as a new FID signal under next available `EXPNO` and process it with `efp` command (Fig. 27).

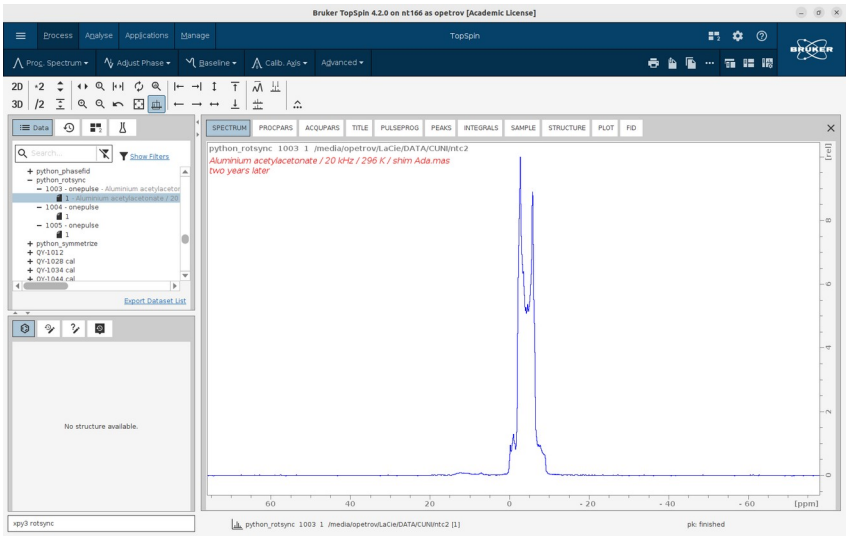

Fig. 25

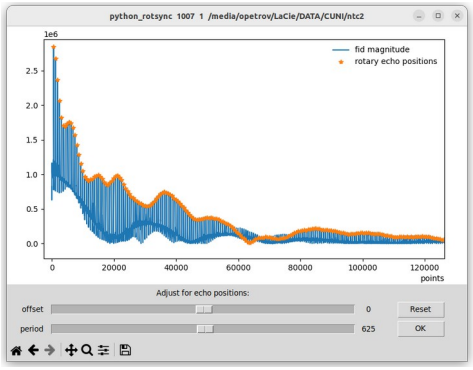

Fig. 26

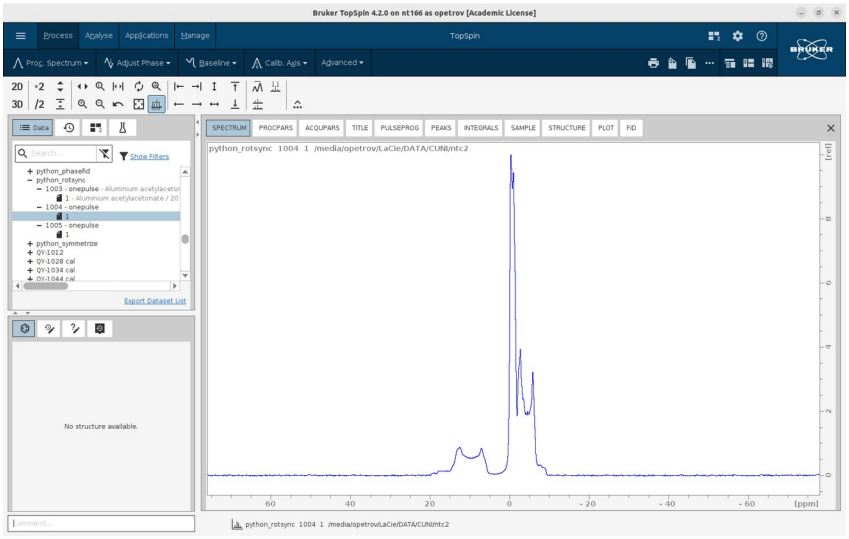

Fig. 27

4.11 rotsync_plus

Identical to `rotsync` except for a more involved procedure of echo quantitation, aiming to better SNR.

4.12 symmetrize

This, again, applies to MAS NMR data, namely to quadrupolar MAS spectra of half-integer spins. The spinning sideband (ssb) manifolds in such spectra may appear highly asymmetric with respect to the central band, even when the first-order quadrupolar interactions prevail meaning a high degree of symmetry. The observed ssb intensity distributions are skewed both positively and negatively, suggesting there being several competing factors of such asymmetry. These factors include magic angle misadjustment, uneven probe response as a function of frequency, a noneligible second-order quadrupolar term, and maybe others. Fig. 28 is an example of ssb distribution with positive skewness (^{27}Al NMR). If you really need it appear symmetric, *e.g.* to see how ssb manifolds from different samples compare, or to fit the first-order quadrupolar model to the ssb pattern, or simply to present the spectrum at its best (*i.e.*, without instrument factors engaged), then use this script.

Typing `xpy3 symmetrize` brings about a two-graph window (Fig. 29). On the top graph you can see the original spectra with ssb's marked with asterisks. In red is shown the recognized centerband (cb) area that is kept intact upon symmetrization. You can re-adjust the cb limits with the sliders. The cb recognition relies on a Topspin's auto-integration method (see `int auto`), namely on the integral regions listed in the `intrng` file. The bottom graph shows a symmetrized spectrum with opposite ssb's equalized using their average intensity. (The averaging is hard-coded in line 105 of the script by assigning `mode='mean'`. If needed, this can be changed for `mode='maximum'` in which case the ssb of a greater intensity will be placed on either side of the spectrum.) Clicking `OK` will return the symmetrized spectrum to Topspin (Fig. 30).

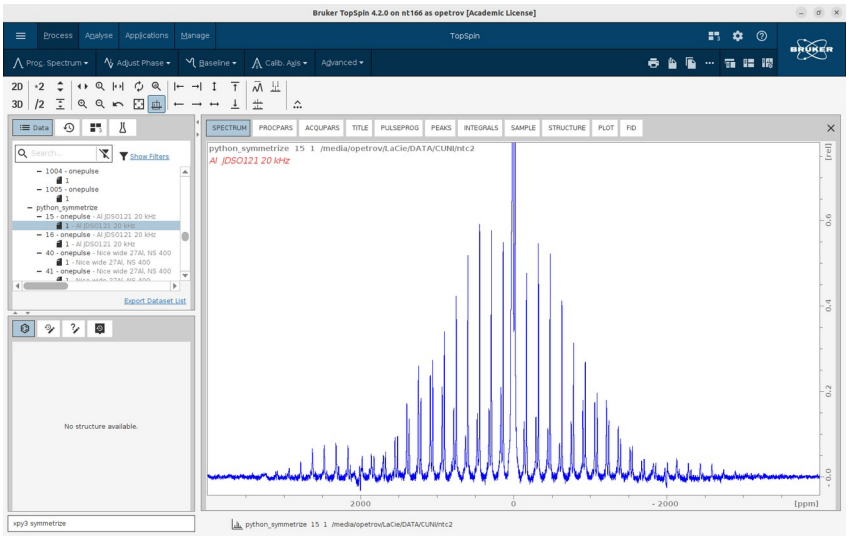

Fig. 28

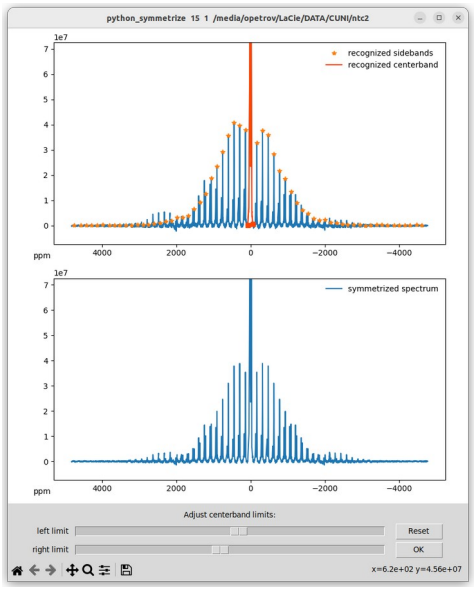

Fig. 29

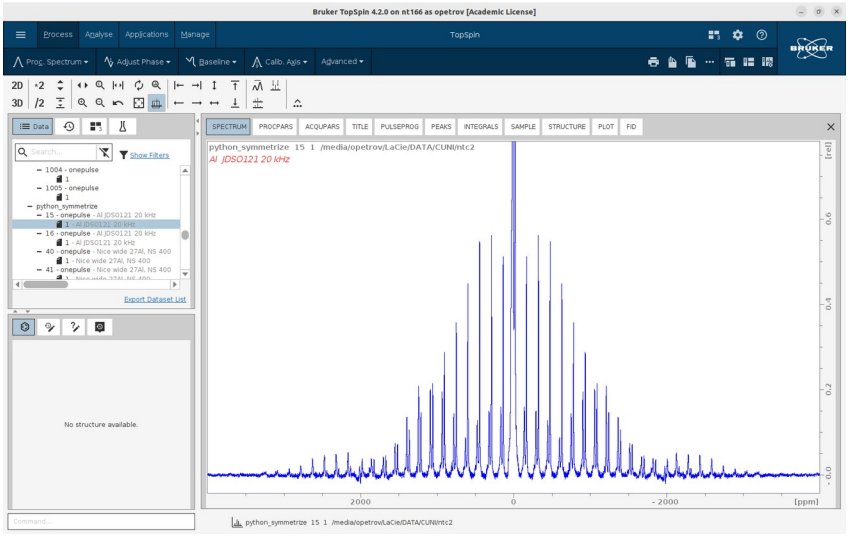

Fig. 30

4.13 t1t2

As the name suggests, `t1t2` is similar in functions to Topspin's T1T2 relaxation analysis toolbox. It integrates a series of spectra measured at varying delays (pseudo 2D data) over an interactively selected region and fits an exponential model to the integral vs delay plot. Unlike the Topspin's T1T2, it works only with processed data, not with FIDs. Besides, it doesn't seek the biggest peak in the integration region, plotting only the integral itself. The relaxation models used are mono-, two-, and stretched exponential. The relaxation delays are read from a VDLIST file, sorted in ascending order. If a VDLIST file is not found, the integrals are plotted against their index numbers, in which case the fitting is unavailable.

The script has an option for signal denoising based on SVD threshold method, helping to decrease a scatter in points in relaxation curves. Another option, applicable to spectra with non-zero baselines, is an automatic subtraction of a baseline shift contribution from the curve. (Apparently, the latter option needs more testing. Should it fail, use `baseline2` instead to correct the spectra for baselines in the first place.)

Fig. 31 shows a two-graph GUI popped up after typing in a command `xpy3 t1t2` on the dataset shown in Fig. 4 (a T_1 inversion-recovery experiment with 24 delays). The top graph displays superimposed spectra where the user can interactively select the desired region. After selection, a relaxation curve will show up on the lower graph (Fig. 32). The curve is continuously updated upon new selections. In this example, an intrinsic mono-exponential relaxation of the selected peak is obscured by non-zero baseline shift. To fix it, you have to check a `Baseline corrector` checkbox, then click `Fit Me!` to fit a mono-exponential model to the data (Fig. 33).

Clicking a `Save&Quit` button saves both the experimental and the best-fit curve in a text file `<EXPNO>/pdata/1/ct1t2.001`. Note that the annotation text with the best-fit model parameters is not saved, so mind that you write the needed values down before quitting.

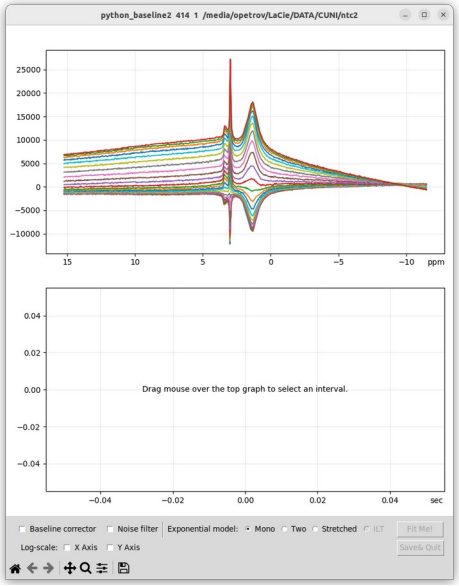

Fig. 31

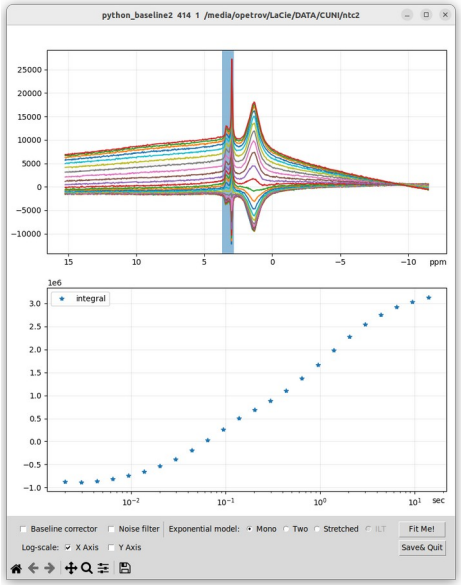

Fig. 32

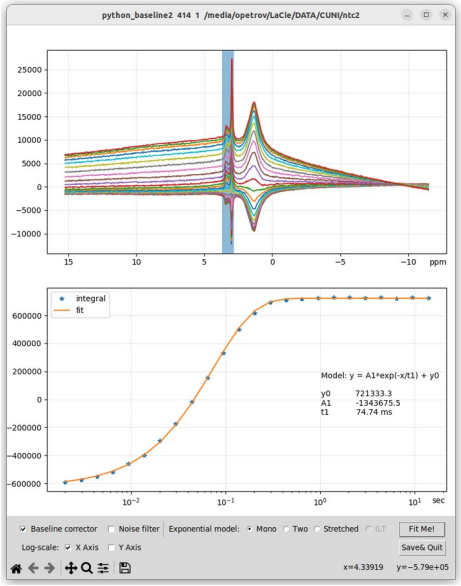

Fig. 33

5 Final remarks

The Bruker API for Python added to recent versions of Topspin [1] is a tempting option for those who used to write standalone Python scripts for their NMR data processing. The `xpy3-tools` is a small collection of such formerly standalone scripts turned into Topspin add-ons, decorated in one style. The tasks performed by the `xpy3-tools` may seem too diverse or, on the other hand, too specific, and therefore make an impression that `xpy3-tools` will not find its use. But then AU programs and Jython scripts from the Topspin library are diverse too, yet they are used by many. Anyway, this version of `xpy3-tools` is just a base for a greater collection of scripts. The experienced NMR users who tend to write their own scripts might consider it a template / building blocks / complement to their own efforts.

6 References

[1] https://www.bruker.com/en/products-and-solutions/mr/nmr-software/topspin/topspin-python-interface.html

[2] https://github.com/derb12/pybaselines

[3] https://github.com/WFP-VAM/vam.whittaker/

[4] Wang, K.C. et al. Distribution-Based Classification Method for Baseline Correction of Metabolomic 1D Proton Nuclear Magnetic Resonance Spectra, *Analytical Chemistry* **85** (2013) 1231-1239.

[5] Eilers, P.H. A perfect smoother, *Anal. Chem.* **75** (2003) 3631–3636.

[6] Kolyagin, Y.G. *J. Phys. Chem. Lett.* **7** (2016), 1249–1253

[7] Gavish, M. and Donoho D. L. The Optimal Hard Threshold for Singular Values is 4/√3, *IEEE Trans. Inf. Theory* **60** (2014) 5040–5053

[8] Petrov, O.V., The Use of Self-Adaptive Principal Components in PCA-based Denoising, *J. Magn. Reson.* **371** (2025) 107824

[9] https://github.com/sekro/pyicoshift

[10] https://contourpy.readthedocs.io/en/v1.1.0

[11] Laurent, G. et. al, Denoising Applied to Spectroscopies – Part I: Concept, *Appl. Spectrosc. Rev.* **54** (2019) 602–630

[12] https://doi.org/10.1039/9781849737531