ASYMEXFIT

ASYMmetrical EXchange FITting

Toolbox for MATLAB to fit the nuclear magnetic resonance spectral lineshape for the general case of two-site chemical exchange

© 2018, Václav Římal, version 2.4
Faculty of Mathematics and Physics, Charles University

Contents

1. Conditions of use

Author of this toolbox is Václav Římal from Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic. You are welcome to report bugs or ask questions about the toolbox at the author's e-mail `vaclav.rimal@mff.cuni.cz`.

Any publication any part of which was based on or created with the help of the Asymexfit toolbox should cite the original paper [1]:

Římal, V., Štěpánková, H., Štěpánek, J., *Analysis of NMR spectra in case of temperature-dependent chemical exchange between two unequally populated sites,* Concepts Magn. Reson. A, **38A,** 117–127 (2011), doi: 10.1002/cmr.a.20214.

The Asymexfit toolbox is free of charge. It may be distributed freely unmodified, but always with a reference to the original download location that is `http://nmr.mff.cuni.cz/`. It is not permitted to utilize any part of the software in commercial products without prior written consent of the author. You may modify any part of the toolbox, but if you wish to distribute the toolbox with your modifications, make sure you have properly commented all the modifications in the source code and insert a notice about your modifications to the header of the main file asymexfit.m. Preserve all the copyright notes in each file of the toolbox.

The Asymexfit toolbox must always be distributed with all its components (listed in the section Files), even if you have modified any part of the toolbox or added a file to it.

The Asymexfit toolbox is provided without warranty of any kind, either expressed or implied, including but not limited to the implied warranties or conditions of mechantability or fitness for a particular purpose. In no event shall the author be liable for any general, special, incidental, indirect, or consequential damages of any kind, or damages resulting from loss of use, data, or profits arising out of or in connection with the use or performance of this software.

2. Installation

To be able to run the Asymexfit toolbox, you must have MATLAB[1] installed with the Optimization Toolbox, which is used by Asymexfit (particularly, the `lsqcurvefit` function).

For installation of Asymexfit, unpack the archive `asymexfit.zip` or `asymexfit.tgz` to a directory of your choice. Then, add this directory in the MATLAB path list (in the MATLAB menu File/Set Path...).

[1]for the support of classes defined in Asymexfit, MATLAB of version at least 7.6 is required

If you saved Asymexfit in the MATLAB toolbox directory, after actualization of the files it is recommended to run the `rehash toolbox` command in the MATLAB command window to ensure indexing the functions properly.

3. Files

The toolbox Asymexfit contains the directories `@EyringData`, `@NMRspectra`, `@populationData`, `@TDdata`, and `example`. The following files are stored in the root directory:

- `readme.pdf`
- `asymex.m`
- `asymexDisp.m`
- `asymexfit.m`
- `asymexopt.m`
- `axisinterpol.m`
- `calcconflimits.m`
- `calcsigma.m`
- `cell2matEmpty.m`
- `chi2red.m`
- `confidence.m`
- `Contents.m`
- `deltaGact.m`
- `deltaGTm.m`
- `echotime.m`
- `emptyasymexfitstruct.m`
- `exportpars.m`
- `extractchi2.m`
- `extractpar.m`
- `Eyring.m`
- `EyringFit.m`
- `getconstpar.m`
- `loadBruker.m`
- `Lorentz.m`
- `LorentzPower.m`
- `makecolumnpars.m`
- `meanfinite.m`
- `MHz.m`
- `noiseerr.m`
- `populationA.m`
- `populationABC.m`
- `populationAfit.m`
- `populationAUnimol.m`
- `recalctotalsigma.m`
- `refillasymexopt.m`
- `resolvecouplingpatterns.m`
- `restorecolumnpars.m`
- `scale.m`
- `sortbynames.m`
- `spc.m`
- `spccomplex.m`
- `spcJpatterns.m`
- `spcJpatternsSepar.m`
- `spcSameInt.m`
- `spcSameIntSepar.m`
- `spcSameM0.m`
- `spcSameM0Separ.m`
- `spcSepar.m`
- `spcSlowEx.m`
- `spcSlowExJpatterns.m`
- `spcSlowExJpatternsSepar.m`
- `spcSlowExSepar.m`
- `stdfinite.m`
- `storeDiary.m`
- `sumfinite.m`
- `varyT2.m`
- `varyw.m`
- `weighteddifference.m`

The directory `@EyringData` contains these files:

assignfunc.m	getvaluesfrom.m
copyobj.m	plot.m
EyringData.m	

The directory `@NMRspectra` contains these files:

actualizepars.m	importresults.m
addpeaks.m	loadobj.m
addpeakstopars.m	makespectralrangenan.m
allfittedcurves.m	next.m
checkbounds.m	NMRspectra.m
clearallbackups.m	plot.m
createbackup.m	plotAll.m
createplottitle.m	plotcpars.m
disp.m	prev.m
estimateerrors.m	recalcallerrors.m
evPropChangedCallback.m	removepeaks.m
exportpars.m	renamepeak.m
fit.m	replot.m
getparsfrom.m	setppmxaxis.m
goto.m	swapppmsign.m
gotofirst.m	undo.m
gotolast.m	

The directory `@populationData` contains these files:

assignfunc.m	evConcChangedCallback.m
calcDeltaTmfromCovUnimol.m	getvaluesfrom.m
calcTm.m	horzcat.m
copyobj.m	plot.m
deltaGerror.m	populationData.m

The directory `@TDdata` contains these files:

calcDeltaG0fromCov.m	getvaluesfrom.m
deltaGerror.m	horzcat.m
evPropChangedCallback.m	plot.m
fit.m	TDdata.m

The directory `example` contains these files:

```
echotime.m                    scale.m
getconstpar.m                 T2A.dat
matlab.final                  T2B.dat
matlab.mat                    wA.dat
matlab.orig                   wB.dat
noisespc.dat
```

4. Theory

Here is a brief introduction to the theory used to derive the spectral lineshape. The theoretical background is described in more detail in the original paper [1].

Suppose this scheme of chemical exchange between two sites, A and B:

$$\mathrm{A} \underset{k_\mathrm{B}}{\overset{k_\mathrm{A}}{\rightleftharpoons}} \mathrm{B}, \tag{1}$$

where k_A and k_B are the exchange rates. Relative populations of both sites will be denoted p_A and p_B.

The Bloch equations for nuclear magnetization can be easily modified to suite the case of two-site chemical exchange. These modified equations are often called Bloch–McConnell equations:

$$\frac{\mathrm{d}M_x^\mathrm{A}}{\mathrm{d}t} = \gamma\left(\boldsymbol{M}^\mathrm{A}\times\boldsymbol{B}^\mathrm{A}\right)_x - \frac{M_x^\mathrm{A}}{T_2^\mathrm{A}} \quad -k_\mathrm{A}M_x^\mathrm{A} + k_\mathrm{B}M_x^\mathrm{B} \tag{2}$$

$$\frac{\mathrm{d}M_y^\mathrm{A}}{\mathrm{d}t} = \gamma\left(\boldsymbol{M}^\mathrm{A}\times\boldsymbol{B}^\mathrm{A}\right)_y - \frac{M_y^\mathrm{A}}{T_2^\mathrm{A}} \quad -k_\mathrm{A}M_y^\mathrm{A} + k_\mathrm{B}M_y^\mathrm{B} \tag{3}$$

$$\frac{\mathrm{d}M_x^\mathrm{B}}{\mathrm{d}t} = \gamma\left(\boldsymbol{M}^\mathrm{B}\times\boldsymbol{B}^\mathrm{B}\right)_x - \frac{M_x^\mathrm{B}}{T_2^\mathrm{B}} \quad +k_\mathrm{A}M_x^\mathrm{A} - k_\mathrm{B}M_x^\mathrm{B} \tag{4}$$

$$\frac{\mathrm{d}M_y^\mathrm{B}}{\mathrm{d}t} = \gamma\left(\boldsymbol{M}^\mathrm{B}\times\boldsymbol{B}^\mathrm{B}\right)_y - \frac{M_y^\mathrm{B}}{T_2^\mathrm{B}} \quad +k_\mathrm{A}M_y^\mathrm{A} - k_\mathrm{B}M_y^\mathrm{B} \tag{5}$$

and the coloured terms describe the flux of magnetization from one site to the other. This set of linear differential equations can be solved to obtain the time evolution of the nuclear magnetization in transverse plane for both sites, which gives after the Fourier transform the spectral lineshape:

$$S(\omega) = M_0\frac{k_\mathrm{A}+k_\mathrm{B}+p_\mathrm{A}\alpha_\mathrm{B}+p_\mathrm{B}\alpha_\mathrm{A}}{\alpha_\mathrm{A}\alpha_\mathrm{B}+k_\mathrm{A}\alpha_\mathrm{B}+k_\mathrm{B}\alpha_\mathrm{A}}\,, \tag{6}$$

where

$$\alpha_\mathrm{A,B} = \frac{1}{T_2^\mathrm{A,B}} - \mathrm{i}(\omega_\mathrm{A,B}-\omega). \tag{7}$$

Here, ω means the angular frequency axis, ω_A and ω_B the resonance frequencies of each site, and T_2^A and T_2^B their transverse relaxation times. M_0 is the size of total nuclear magnetization in equilibrium.

If the system is found in thermal equilibrium, the exchange rates and populations are no more independent; the following expression is valid:

$$p_\mathrm{A} k_\mathrm{A} = p_\mathrm{B} k_\mathrm{B}. \tag{8}$$

Thus, together with the identity

$$p_\mathrm{A} + p_\mathrm{B} = 1, \tag{9}$$

only two quantities of these four are independent. We select k_A and p_A for further treatment, the other two being calculated from them.

The spectral lineshape $S(\omega)$ is valid for any case of two-site chemical exchange, with no restrictions on the population ratio and, generally, the transverse relaxation times are different for both sites, as well as their chemical shifts (represented in the angular-velocity scale as ω_A and ω_B).

The final goal is to determine the quantities describing the exchange process, i. e., values of k_A and p_A. In order to be able to apply the formula for $S(\omega)$ directly to the fitting procedure, number of free parameter should be reduced, especially when only one peak for the exchanging pair of nuclei is visible in the NMR spectrum. In that case, one should estimate the properties of isolated sites A and B (their T_2's and chemical shifts). These values are then employed as fixed.

5. Asymexfit toolbox overview

This section is not meant to give all the information you need to use the Asymexfit toolbox, but is rather an overview of its components. To work efficiently, try the example below and look at each M-file you are going to use and follow the instructions provided in the comments therein.

Help on any function, class or method included in Asymexfit can be obtained directly in the MATLAB environment by:

- pressing the F1 key after typing the name of a component of Asymexfit or after selecting it, directly in the MATLAB command line or in the MATLAB file editor;
- the command `help` followed by the name of a component of Asymexfit; this displays the help into the command line;
- the command `doc` followed by the name of a component of Asymexfit; this opens a window in the MATLAB help browser.

For getting started, the most useful commands are perhaps these:

```
doc asymexfit;
doc NMRspectra;
```

5.1 Classes

The Asymexfit toolbox uses object-oriented programming technique since version 2.0. The functionality without classes is preserved, too.[2] However, using classes makes the work much more simple and efficient by keeping the interconnected variables in a group and by radically shortening the commands user is supposed to type in the MATLAB command line. This manual deals only with the object-oriented part of the Asymexfit toolbox; for hints on no-classes approach, you are welcome to read the readme file of version 1.1 or lower and the comments within the function M-files themselves.

Asymexfit provides four classes, definitions and methods of which are stored in appropriate subdirectories. The classes defined in the toolbox are:

- `NMRspectra`: provides interface for handling the spectra, plotting them together with simulated spectra, and fitting;
- `TDdata`: abstract class providing functionality for evaluation of the melting profiles;
- `populationData`: a subclass of the `TDdata` class evaluating the temperature dependence of relative population of an exchanging site, thus yielding the enthalpy and entropy changes and Gibbs free energies and transition temperatures (as their midpoints) calculated from them;
- `EyringData`: a subclass of the `TDdata` class evaluating the temperature dependence of exchange rates, thus yielding the Gibbs free energies of activation.

5.2 Calculating a spectrum

The Asymexfit toolbox is designed to perform least-square fitting of a model describing one-dimensional nuclear magnetic resonance (NMR) spectra to the experimental data. The main purpose is to analyse spectra in which the lineshapes are modified by chemical exchange. More specifically, the lineshape of spectrum influenced by a two-site chemical exchange between two unequally populated sites is calculated by function `asymex`. By fitting this curve to experimental spectra,

[2]In this version, the classes still use inside them the functions from outside of the defined classes. The classes provide only an interface which makes everything more elegant.

you can obtain relative populations of the two sites and the exchange rates as well. Originally, this software was developed to analyse the temperature dependent spectra of DNA double-helix covering the duplex melting region, but can be used to variety of systems where chemical exchange plays an important role [2]. However, Asymexfit is designed to be most useful when the exchange is in the intermediate to fast regime, so the two exchanging peaks are not separated.

The toolbox can be used for fitting NMR spectra without chemical exchange, too. The Lorenzian curve is calculated by function `Lorentz`, you can also use the Lorentz curve to the power of a general exponent by function `LorentzPower`.

Both Lorentz and exchange curves are calculated in phase sensitive mode, so you can optimize the phase angle of the spectra as well during the fitting procedure, which is automatically done by the `fit` method of the class `NMRspectra`. In addition, a linear correction of the spectral baseline is implied. These parameters are always stored in the last line of the parameter matrix, as specified in detail within the comments in the M-files.

Other models of the spectral lineshapes can be implemented. All you have to do is storing the user-supplied functions to the fields `options.func` and `options.funcSepar` of your object of the `NMRspectra` class. You can use predefined functions `spcSameInt[Separ]` or `spcSameM0[Separ]` for forcing selected peaks to have same intensities, variants of `spcJpatterns` for more complicated weak-coupling schemes, or any user-implemented function. In this manner, you can – in principle – fit any model to your data (so it even needn't necessarily be NMR spectra).

5.3 Spectral fitting

Before starting the analysis, prepare your experimental data and load them together with the frequency axis into MATLAB. If you have the data in Bruker format, you can use the `loadBruker` function to convert them into MATLAB. Create file `MHz.m` (most easily and reliably by copying it from the Asymexfit directory and modifying it), if your basic frequency is other than 500 MHz to override the default setting. Create also files `scale.m` and `echotime.m` to suit your experimental conditions in the same manner. To fit the spectra with chemical exchange, supply function `getconstpar.m` which returns static parameters describing the isolated sites.

Fitting occurs for one spectrum at a time by the method `fit`, but a serial fitting of selected spectra can also be run by providing extra arguments. All parameters and options (including really everything which is in the `options` property, even any user-added fields, which can be also very useful) are stored for the respective spectrum. These can be loaded to another spectrum by the `getparsfrom` method.

5.4 Error estimation

After having done the fit of your experimental spectra, you would probably like to estimate the errors of your results. Asymexfit provides tools for this purpose: varying static parameters of the individual sites (the chemical shifts and transverse relaxation times) and adding Gaussian noise to the optimised curve and repeating many times the fitting procedure with so-created artificial datasets [3]. All this can be done by the `estimateerrors` method of the `NMRspectra` class.

5.5 Evaluation of melting profiles

If you analysed a series of temperature-dependent spectra and you succeeded in determining the values of relative populations and exchange rates together with their errors for each temperature, you can pass the results to the classes `populationData` and `EyringData`. These are designed to yield the entropy and enthalpy changes of the exchange process and of its activation, ΔH, ΔS, $\Delta H^{\dagger}$, and $\Delta S^{\dagger}$. From these quantities and their covariance matrix, the Gibbs free energy difference of the transition or activation, $\Delta G_{310\,\mathrm{K}}$ or $\Delta G^{\dagger}_{310\,\mathrm{K}}$, and the transition temperature of the DNA double-helix melting or another process under study, T_m, and their errors are calculated.

If the spectral analysis was used to evaluate the chemical shifts only instead of p_{A}, the `populationData` class can also be used to reveal the thermodynamic parameters. In this case, a proper model function describing the temperature dependence must be stored in the `func` property and other settings must be also optimised before the fit.

6. Example

In the folder `example`, you can find all you need to perform the fitting of spectra with chemical exchange. Loading `matlab.mat` into MATLAB will create variables prepared for the analysis, containing the experimental spectra, too. These are parts of 500 MHz 1D ^{1}H NMR spectra of oligodeoxynucleotide d(GATGCATC)$_2$, in dependence on temperature (from lower to higher). The fraction of duplexes decreases with increasing the index of spectrum, and the exchange rate increases in this direction (because the temperature grows).

6.1 Preparing for fit

In MATLAB workspace with the directory `example` as the current path, you can run the following commands:

`load;`	*Load workspace*
`s=NMRspectra(ppm,spectra);`	*Create an object of the class NMRspectra*
`s.T=T;`	*Add temperature axis into the object*
`s.addpeaks([5 0]);`	*Add five peaks with chemical exchange*
`s.peaknames=names;`	*Attach names to all peaks*
`s.pars=guess;`	*Set initial parameters*
`s.mult(5)=2;`	*Set multiplicity of C8H6 to 2 (doublet)*
`s.options.min=lobound;`	*Set lower bounds for fitting*
`s.options.max=hibound;`	*Set higher bounds for fitting*
`s.plot;`	*Plot with better scaling of the axes*

Now, all the variables needed are stored in the object `s` and the figure window should display the first spectrum together with spectrum calculated from the parameters in the property `pars` from the object as in Fig. 1. The last peak is a doublet (cytosine H6 resonance), whereas the others are singlets. The frequency axis is in ppm units increasing from right to left, as is common in the high-resolution NMR spectroscopy.

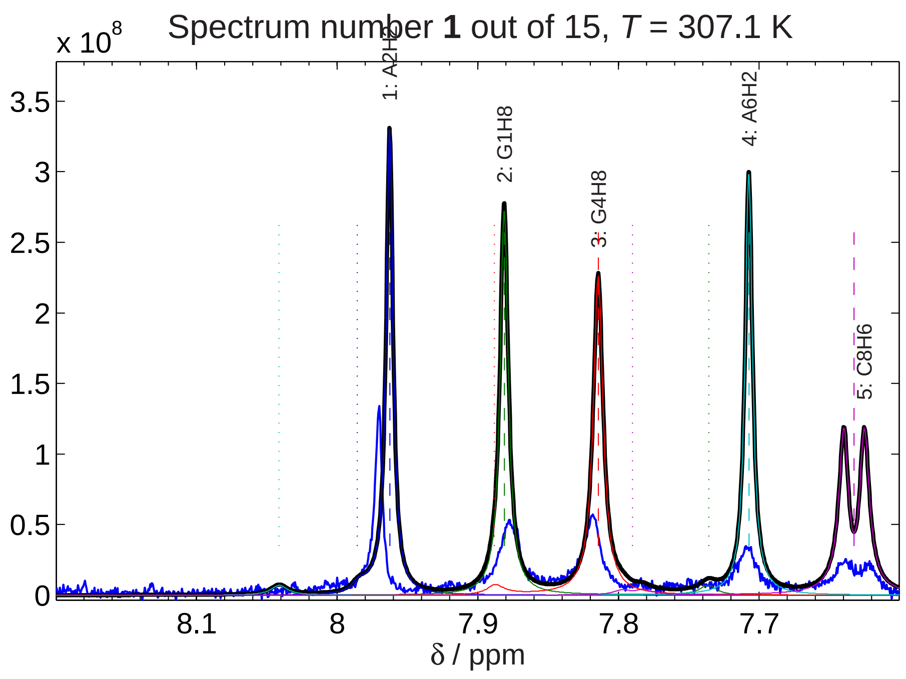

Figure 1: First spectrum with initial guess

Throw an eye on the values in variable `guess` (or in the property `s.pars`), which are going to be used as the initial guess for fit. For exchanging peaks, in each row there are (from left to right): the intensity M_0, the molar duplex fraction

p_A, the first-order dissociation rate constant k_A in s^{-1}, and the scalar coupling constant J in Hz. Full meaning of these quantities is explained in Section 4. Further explanation of the values in the parameter array used to calculate the spectrum goes in the header of the `spc.m` file (which can be viewed directly in the Matlab Editor, by typing `doc spc` in the Matlab command line, or by pressing the F1 key after selecting `spc` written in the Matlab command line). Editing of any of the parameter values in `s.pars`, no matter if directly in the variable editor or through the command line, immediately updates the figure.

6.2 Fitting!

Now you are really ready for fit. Use the method `fit` of the object `s` as easily as follows:

```
s.fit;
```

This command fits the actual spectrum (i.e., the first one) and stores the results to the `fitResults` property of the object. The command also produces this text in the MATLAB command line:

```
Creating 2th backup of 1th spectrum
asymexfit >> *** Fitting spectrum number 1 ***
asymexfit, sp. 1 >> Original chi2 = 2.21149730e+018
Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.
asymexfit, sp. 1 >> Actual chi2 = 5.52497198e+015
asymexfit, sp. 1 >> Rel. change in chi2 = -99.75 %
asymexfit, sp. 1 >> Position of the most changing parameter: 6  1
asymexfit, sp. 1 >> Its original and new value: 0      7.3232
asymexfit, sp. 1 >> Position of the most changing parameter
 excluding last row: 1  3
asymexfit, sp. 1 >> Its original and new value: 1       149.047
```

and plots the results in the current graphics window (Fig. 2).

To switch to next spectrum in the series, just use the method `next`[3]. In this way, you can continue and repeat the procedure for all the spectra. Every time you switch to another spectrum, the parameter values are stored into the property `pars`, etc. All results together with all the options for the fits are stored in the `fitResults` property.

[3]to go back, use the `prev` method; to switch to any spectrum, use `goto`. For details, see the file `NMRspectra.m`

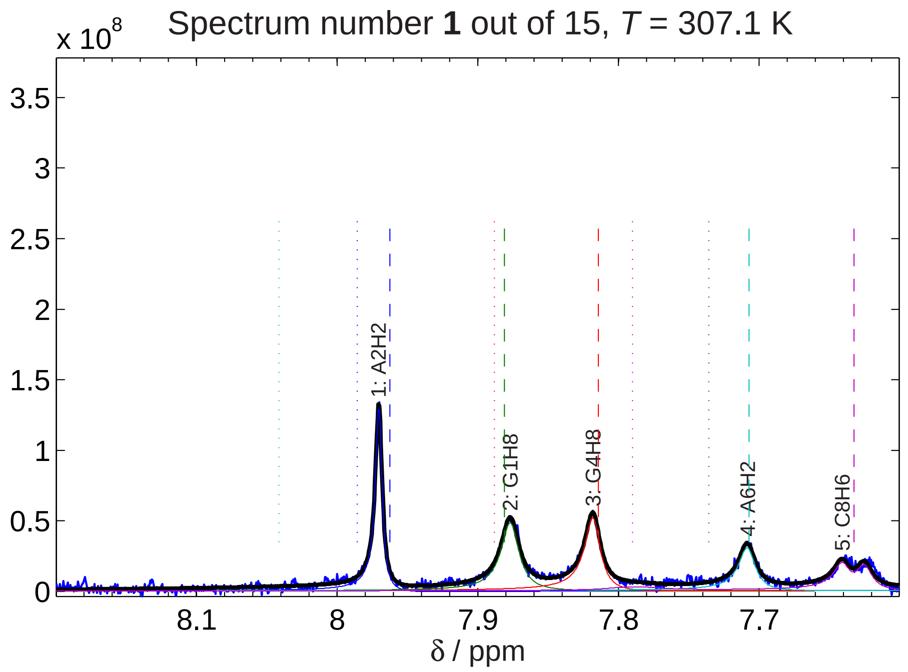

Figure 2: Final view after the first spectrum is fitted

Upon any change in peak number and before each fitting, a backup is created and this works for each spectrum independently. This allows you to go back to a specific point of your work by the method `undo`.

Within all the functions of Asymexfit, error messages are marked with `E:`, while warnings with `W:`. Notifications are only beginning with the name of function producing it and a `>>`.

6.3 Error estimation

There are three main sources of errors in the lineshape analysis in the case of chemical exchange: estimation of chemical shifts, estimation of transverse relaxation times (T_2^*) of the isolated sites undergoing the exchange, and spectral noise. Influences of these error sources are evaluated by the method `estimateerrors` of the `NMRspectra` class.

Inspect the files `T2A.dat, T2B.dat, wA.dat,` and `wB.dat`. Structure of them is described in the headers of `varyw.m` and `varyT2.m`. In the file `noisespc.dat`, spectral region from 12.0 ppm to 8.5 ppm with no resonances is stored for each temperature.

After running the `estimateerrors` method (you can supply the indeces of spectra to be evaluated), the results are stored into the fields `T2variations`, `T2sigma`, `wvariations`, `wsigma`, `conflimits`, and `conflevels` of the `fitResults` property of the object `s`.

Yes, some errors occured, but don't worry – we have just been several times trying to put fairly wrong estimates of ω_A and ω_B, so this is not the problem.

The outputs of `estimateerrors` have been stored into log files, which you will certainly appreciate as the text can be quite long, especially if you fit more spectra in a row.

6.4 Thermodynamic parameters

After all the three steps described in the above subsection, all important results are stored in the property `fitResults` in your `NMRspectra` object. You can now extract all the fitted parameters including the relative fraction of duplexes and exchange rates in dependence on temperature and their errors, together with the peak names, by the `exportpars` method:

```
[pars, spars]=s.exportpars;
```

You can use Asymexfit for analysing the temperature profiles of populations of site A, $p_\mathrm{A}(T)$, and the rate constants, $k_\mathrm{A}(T)$. For the former, the class `populationData` is designed, and the class `EyringData` serves for the latter. Both are constructed from an object of the `NMRspectra` class, say, `s`, as follows:

`p = populationData(s,idx,conc)`	*Create an object of `populationData` class*
`k = EyringData(s,idx)`	*Create an object of `EyringData` class*

The variable `idx` represents a list of peaks for which the data should be loaded and `conc` is the concentration of 2A + B in mM[4]. Alternatively, you can create an empty object by not supplying any arguments to the constructor, and then input the requested values in its properties manually.

Functions used to calculate other quantities and their errors from the input data are stored in the properties `func` and `funcErr`, which are three-membered cell-arrays of function handles. They can be overridden by any user-supplied function. See `doc populationData` and `doc EyringData` for more information.

A special attention deserves the property `equation` of the `populationData` class, because it describes the process under investigation. Currently, three types of process are supported: $\mathrm{A} \rightleftharpoons \mathrm{B}$, $\mathrm{A} \rightleftharpoons \mathrm{B} + \mathrm{B}$ (default), or $\mathrm{A} \rightleftharpoons \mathrm{B} + \mathrm{C}$. This property influences calculation of the equilibrium constant and other quantities, hence gives different results of the fit. After setting the `equation` property to a new value, function used to calculate the equilibrium quantities are changed to respect the new type of process. Note that all the supported schemes are first-order in A, thus the discrimination is unnecessary in the `EyringData` class.

[4]designed as the total concentration of the DNA single strands by default

Since the classes `populationData` and `EyringData` are both derived from the same (abstract) class, `TDdata`, most of the methods are the same. The following text deals only with the `populationData`, but it holds for the `EyringData` class in a similar way, too.

The temperature dependence can be fitted by appropriate function, which is stored in the property `func` in the object. Fit of all the profiles is done in the simplest way by typing

```
p.fit;
```

The optimized values of enthalpy change ΔH and entropy change ΔS are stored in the property `HS` and the plot is actualized (Fig. 3). The estimated errors of the enthalpy and entropy values are stored in the `HSdelta` property.

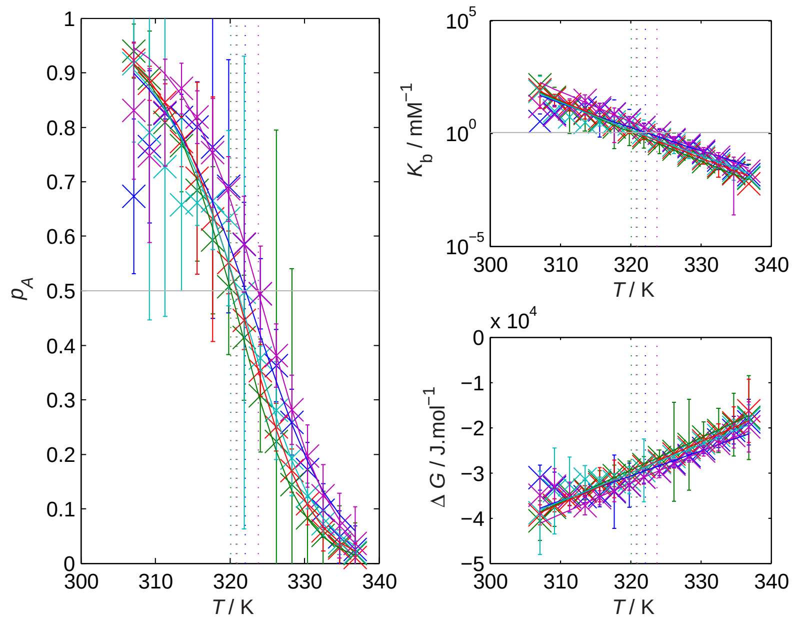

Figure 3: Results of fitting the relative population of site A; the plot was produced by the method `fit` of the `populationData` class. Left: temperature T in K is on the horizontal axis, the duplex population p_{A}on the vertical axis. Top right: semilogaritmic plot of the equilibrium constant of the backward process, K_{b}, versus temperature. Bottom right: plot of the Gibbs free energy difference of the backward process, $\Delta G_,$ versus temperature. Grey horizontal lines represent the state when $p_{\mathrm{A}} = 0.5$, dashed vertical lines are at $T = T_m$ for each profile

7. Changes in versions

7.1 What is new in version 2.4 (released April 12, 2018)

7.1.1 New methods, properties, and functions

- `NMRspectra`:
  - `makespectralrangenan`: method for removing a selected region from the actual spectrum by making the appropriate values NaN;
  - `clearallbackups`: method for deleting all the backups of all spectra in the object;
- `sumfinite`: function providing sum of finite values in an array only.

7.1.2 Major improvements

- `NMRspectra`:
  - `getparsfrom`: extended modality – output to a variable and selection of some peaks only are supported;
  - `plot` and `replot`: shifts of the two exchanging sites are drawn by vertical lines (governed by the plot option, bit code 4);
  - `renamepeak`: only some spectra in the series can be selected for peak renaming;
- `asymexfit`:
  - shifts of the two exchanging sites are drawn by vertical lines (governed by the plot option, bit code 4);
  - only finite values of the spectrum are used for the chi-square calculation, hence NaN values can be used for skipped regions;
- `resolvecouplingpatterns`: phase differences in more complicated multiplets are now calculated when a spin-echo pulse sequence is used for acquisition;
- `spcSameInt` and `spcSameIntSepar`: generalisation for calculating any model spectrum by enabling a function handle to be provided as an extra argument.

7.1.3 Fixed bugs

- `NMRspectra`:
  - in plotting, spectra with empty pars are skipped in order not to get an error;

  - there was a mishandling of prefactor in some spectral calculations and plotting;
  - compatibility with complex spectra provided in plotting;
  - texts are now properly positioned and spaced in plots;
  - `plotAll` repaired for the case when one spectrum only is present in the object;
  - `exportpars` returns all the columns of spectral parameters;
  - `undo` was not reading all fields in options when returning to a backup;

- `TDdata`, `populationData` and `EyringData`:
  - missing labels do not cause errors and the plot title is now displayed without undesired lines around;
  - the constructors respect the proper number of columns in `pars` when loading from a `NMRspectra` object;

- `populationData`:
  - `horzcat` and `getvaluesfrom` do not rewrite the model functions;
  - concentration change forces updating the model functions;

- `asymexfit`: compatibility with complex spectra provided;

- `resolvecouplingpatterns` was expecting exactly three columns of parameters before J, now generalised to `ncolsbeforeJ` (optional argument);

- `Lorentz` ignored the phase differences when provided as an array and not a scalar.

7.2 What was new in version 2.3 (released January 29, 2015)

7.2.1 New methods, properties, and functions

- `TDdata, populationData` and `EyringData`:
  - `HScovar`: new property containing the covariance matrices of enthalpy and entropy calculated from the fit;
  - `plottedidx`: new private property serving for remembering which column of parameters was plotted last time;
  - `calcDeltaG0fromCov`: new static method for calculating the error of the Gibbs free energy difference coming from the fit using the covariance matrix;

- `populationData`:
  - `calcDeltaTmfromCovUnimol`: new static method for calculating the error of the Gibbs free energy difference coming from the fit using the covariance matrix.

7.2.2 Major improvements

- `NMRspectra`:
  - properties `ppm`, `spectra`, and `actual` are made public for reading; however, `actual` property still has private SetAccess;
  - `getparsfrom`: it can be specified which properties should be copied;
  - `plotAll` plots a stacked plot of all spectra with labels and fitted curves;
  - all columns of parameters describing the spectra are exported by method `exportpars` ;
- `TDdata, populationData` and `EyringData`:
  - properties `sG0, HSdelta` and also `sTm` in `populationData` class are now dependent, which means they are recalulated based on other properties, mainly the covariance matrix `HScovar`;
  - after changing parameters in observable properties, the current graph is replotted only for the columns of parameters which were plotted the last time;
  - a title containing the description from the `atomnames` property is added to the graph if only one column of parameters is plotted;
- `asymexfit`: after each fit, the bound offending values are displayed much more compactly.

7.2.3 Fixed bugs

- `NMRspectra`:
  - `getparsfrom` now creates a backup before importing new data;
- `TDdata, populationData` and `EyringData`:
  - calculation of errors of the fitted parameters corrected, bugs were present for cases when some points where not finite in the input data and when the experimental errors were not supplied due to wrong calculation of the number of degrees of freedom;
  - removed Abstract attribute from `LProps` property;

- `populationData`:
  - `func` property was reset to default by `getvaluesfrom` method;
  - loading an object by MATLAB `load` function may crash due to calling still empty property `equation` by `assignfunc` method
- size check of `T2A` and `T2B` arrays corrected in `varyT2` function;
- only columns 1:4 are used for calculating spectra without exchange by functions `spc` and `spcSepar`, now ignoring any following columns in the input parameters.

7.3 What was new in version 2.2 (released July 1, 2014)

7.3.1 New methods, properties, and functions

- `TDdata, populationData` and `EyringData`:
  - `copyobj`: creates a new object by copying another;
  - `getvaluesfrom`: loads all data from one object to another;
  - `HSdelta`: new property containing errors of enthalpy and entropy calculated from the fit;
- `NMRspectra`:
  - `fittedCurve, fittedCurveSepar`: properties containing the model curves calculated from actual parameters;
  - `allfittedcurves`: exports fitted curves for all or selected spectra;
- `spcSlowEx.m, spcSlowExSepar.m`: functions calculating spectra with slow chemical exchange, i.e., chemical shifts of exchanging sites are in fitted parameters;
- `resolvecouplingpatterns.m`, `spcJpatterns.m`, `spcJpatternsSepar.m`, `spcslowExJpatterns.m`, `spcSlowExJpatternsSepar.m`: functions for calculating more complex J-coupling patterns (assuming only weak coupling).

7.3.2 Major improvements

- `NMRspectra`:
  - in `getparsfrom` method, a source object can be specified;
  - in `estimateerrors` method, a bit-coded parameter specifying which error sources should be considered can be passed as an input argument;

- `TDdata, populationData` and `EyringData`:
  - `fit` method calculates errors of the fitted parameters and stores them in the `HSdelta` property;
  - `HS` property is set observable and the plot is redrawn after changing its values.

7.3.3 Fixed bugs

- `NMRspectra`:
  - error during adding peak when only one was added and its name was passed as an input argument as char, not cell array;
  - when adding peak with parameters having more than 4 columns, the input was truncated;
  - plotting ignored the settings in options;
- `TDdata, populationData` and `EyringData`:
  - concatenating objects was not setting `defGuess`, `minBound`, `maxBound`, `equation`, and `cRef` properties;
- `populationData`:
  - error in putting the T_m lines into the plot for an A $\rightleftharpoons$ B process;
  - wrong setting of plot axes in equilibirum constant section.

7.4 What was new in version 2.1 (released March 17, 2014)

7.4.1 New public methods and functions

- `NMRspectra.getparsfrom`: loads all parameters, options and peak names from a selected spectrum to the actual one;
- `NMRspectra.plotcpars`: adds lines corresponding to the chemical shifts of the sites A and B to the current plot;
- `populationData.calcTm:` calculates melting temperature (transition midpoint) according to the equation of the process under study from enthalpy and entropy change;
- `populationAUnimol, populationABC:` functions calculating population of site A from enthalpy and entropy change for the processes A $\rightleftharpoons$ B and A $\rightleftharpoons$ B + C, respectively.

7.4.2 Major improvements

- `NMRspectra`:
  - fitting performance improved (1): internally, parameter array is converted to a column vector (`makecolumnpars`), skipping unused fields, before the fit, and restored after the fit (`restorecolumnpars`); directed by the field `makeColumnPars` in options; as the result, fitted function is called fewer times: when `MaxIter` value is limiting, time spent by fitting can be shortened by 11 %; when `MaxFuncEval` value is limiting, about 8 % more iterations can be achieved during the same time;
  - fitting performance improved (2): using `refillasymexopt` function only when necessary; the time spent by fitting can be shortened by 25 %;
  - `addpeaks`: supplementary arguments can be added when calling this method, containing parameters for new peaks;
  - `fit`: optionally, `maxfit` can be supplied as an argument to this method;
  - after selecting other spectrum, the plot axes stay unmodified;
  - when plotting, a title with spectrum number and temperature is added;
- `TDdata`:
  - each temperature profile is plotted by the same color every time;
  - negative values in logarithmic plots are not displaying errors in the command line;
  - fitting method `fit` accepts different order of arguments – now more logical;
- `populationData, EyringData`:
  - plotting improvement: three subplots are created; lines representing important values are drawn;
  - generalization of plotting and fitting (1): new properties `func` and `funcErr` added, which can be overridden by the user; both contain three functions used in plotting and calculations, see help for details;
  - constructors can be called with empty arguments;
- `populationData`:
  - generalization of plotting and fitting (2): `equation` property added, which describes the process type;
  - generalization of plotting and fitting (3): plotting equilibrium constant from left to right or from right to left (default) can be selected by the field `whichKtoplot` in options;

  - new dependent properties `Kf` and `Kb`, equilibrium constants for forward and backward reaction, respectively, calculated from `pars` property by function specified in the property `func`;
  - new property `cRef` by which equilibrium constant is multiplied in van't Hoff equation; its default value is 55556 mM to be compatible with previous versions.

7.4.3 Fixed bugs

- `NMRspectra`:
  - error was encountered when the series of spectra was not complete, i.e. some of them were missing or were not with exchange in the beginning of the series;
  - run-time errors in the `errorestimation` method were caused by missing argument `makeboundsinf` in the header of `extractpar` function;
  - using `calcsigma` in varying chemical shift values (`varyw`), which was already implemented in `varyT2`;
- `TDdata`:
  - `options` property changed from protected to public;
  - `G0` property changed to dependent, is calculated from the `HS` property;
  - chi-square calculation caused error when no fitting has been performed yet;
  - `evPropChangedCallback` method and `LProps` property changed from private to protected;
- `populationData, EyringData`:
  - not plotting errorbars and not looking for outliers when no error values are supplied;
  - all workspace active when constructing an object was saved, now this is suppressed and works more correctly by assigning functions by the `assignfunc` method;
- `populationData`:
  - `Tm` property changed to dependent, is calculated from the `HS` property using the `calcTm` method;
  - `evConcChangedCallback` method changed from private to protected;
- `cell2matEmpty` was missing in the asymexfit directory (used in the constructors of `populationData` and `EyringData` classes).

7.5 What was new in version 2.0 (released November 20, 2013)

7.5.1 New classes

- `NMRspectra`: Class for handling and fitting NMR spectra and model functions.
- `TDdata`: Abstract class for plotting and fitting parameters describing chemical exchange.
- `populationData < TDdata`: Class for plotting and fitting the temperature dependence of relative populations.
- `EyringData < TDdata`: Class for plotting and fitting the temperature dependence of exchange rates

7.5.2 New functions

- `axisinterpol.m`: linear interpolation of data to a new axis.
- `calcconflimits.m`: calculates confidence limits from Monte Carlo noise simulations which are stored in the structure fit.
- `calcsigma.m`: calculates errors from T2 or w variations.
- `emptyasymexfitstruct.m`: creates empty structure containing the fields returned by asymexfit.
- `exportpars.m`: exports table of results and errors from structure fit obtained by asymexfit together with names of the peaks in the head of the table, if specified.
- `spcSameInt.m`, `spcSameIntSepar.m`, `spcSameM0.m`, and `spcSameM0Separ.m`: these functions return the NMR spectra with common integral intensity for selected peaks without chemical exchange (`spcSameInt.m` and `spcSameInt-Separ.m`) or for all peaks with chemical exchange (`spcSameM0.m` and `spc-SameM0Separ.m`).
- `sortbynames.m`: sorts columns of data according to a list of names (does not sort the names alphabetically).
- `weighteddifference.m`: weighted difference of two arrays y1 and y2, argument sigma being the weight.

7.5.3 Major improvements

- calculating of the error of the Gibbs free energy of activation is newly internally done by the `deltaGTm` function (`deltaGact.m`).

- new possibility of automatic recursion while refining the step in the ΔH and ΔS axes when estimating the error of ΔG or $\Delta G^{\dagger}$ (`deltaGTm.m` and `deltaGact.m`).
- outliers (points further than 3σ from the fit) are emphasised in the graph and a short notice is displayed in the command line (`EyringFit.m` and `populationAfit.m`)

7.5.4 Fixed bugs

- subfunction `plotExpFit` was generating an error when plotting of spectrum was not set in asymexfit options (`asymexfit.m`).
- use of the `fmincon` function from the Optimization toolbox is deprecated and replaced by the function `lsqcurvefit` (`EyringFit.m` and `populationAfit.m`).
- data falling to the bounds of the fit are not replaced by +Inf or -Inf values when generating error reports, thus these data are also used to calculate the errors (`noiseerr.m, varyT2.m`, and `varyw.m`).

7.6 What was new in version 1.1 (released May 10, 2013)

7.6.1 New functions

- `extractchi2.m:` extracts values of chi-square from the results of fitting for each spectrum.
- `meanfinite.m:` returns mean values of only finite values (ignoring NaN and Inf values).
- `recalctotalsigma.m:` adds field .total into structure with deviations and stores there the square root of squares of partial deviations from the fields .w, .T2 and .noise.
- `refillasymexopt.m:` refilling the structure with options by adding missing fields with default values.
- `storeDiary.m:` stores the command line output generated by the diary on command in the diary file into a new file `<name>N.log`, where N is number appended if necessary to avoid rewriting existing file, and ends the logging by diary off command.

7.6.2 Major improvements

- New options .prefactor and .prefactorFunc enable multiplying of the parameter array field-by-field by output of user-provided function specified in

.prefactorFunc. Useful when there are large differences of magnitudes of the parameter values (`asymexfit.m, asymexopt.m, spc.m`).

- Command line reports made more comfortable to read (`asymexfit.m`).
- Extracting parameters also from arrays with varying numbers of peaks now enabled (`extractpar.m`).
- Eyring plot now with errorbars if provided (`EyringFit.m`).
- Fit boundaries can be supplied as input arguments (`EyringFit.m, populationAfit.m`).
- Default values for confidence levels specified (`noiseerr.m`).
- Average calculated only from finite values (`varyT2.m`).

7.6.3 Fixed bugs

- constpar was created empty even when submitted as input argument (fit without exchange, `asymexfit.m`)
- corrected swapped languages in reports (`asymexfit.m`)
- checking attained boundaries was faulty for negative values (`asymexfit.m`)
- corrected entropy units (`deltaGTm.m, deltaGact.m`)
- wrong determination if all values are NaN corrected (`extractpar.m`)
- control if a file exists corrected (`loadBruker.m`)
- points and lines have the same colors (`populationAfit.m`)

8. References

[1] Římal, V., Štěpánková, H., and Štěpánek, J., *Analysis of NMR Spectra in Case of Temperature-Dependent Chemical Exchange Between Two Unequally Populated Sites*, Concepts Magn. Reson. Part A **38A** (2011), no. 3, 117–127, DOI 10.1002/cmr.a.20214.

[2] Kozic, J., Novák, Z., Římal, V., Profant, V., Kuneš, J., and Vinšová, J., *Conformations, equilibrium thermodynamics and rotational barriers of secondary thiobenzanilides*, Tetrahedron **72** (2016), no. 17, 2072–2083, DOI 10.1016/j.tet.2016.02.035.

[3] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Modeling of Data*, In: Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, 2nd ed., 1992, ISBN 978-0521431088, ch. 15, pp. 656–706.