OXSA Fitting Guide v1.0

10th July 2017

Chris Rodgers – christopher.rodgers@cardiov.ox.ac.uk
Lucian Purvis – lucian.purvis@cardiov.ox.ac.uk

There are two main functions that can be called to fit a spectrum. The first is `AMARES.amares`, and the second is `AMARES.amaresFit`.

`amares` pre-processes the spectrum by finding the offset and performing DC correction. It then passes the spectrum to `amaresFit` and then post-processes the results. Spectra can be passed to `amares` as a time-domain signal, or a frequency domain spectrum. `Example_phantomData` shows `amares` being called.

`amaresFit` is the main fitting code. It phases the spectra, fits them, and outputs the values of and Cramer-Rao lower bounds on the fitted chemical shift, linewidth, amplitude and phase of each of the peaks. The data must be passed to `amaresFit` as a complex time-domain FID. `Example_fitSimSpec` shows `amaresFit` being called.

Name/value pairs

The required variables for the OXSA AMARES functions are inputted by replacing each field with the chosen value. Any additional option are inputted using name/value pairs. These are processed by the `processVarargin` function to give an options structure. The pairs are passed in at the end of the function call as e.g. `AMARES.amares(spec, …, 'fieldName', value)`.

Input for `amares` and `amaresFit`

```
Results = AMARES.amares(spec, instanceNum ,voxelNum,
beginTime, expOffset, pk, showPlot);
```

1) Spec

   A structure with the following fields:

   a) Spectra/signals

      A cell array of the complex data in either frequency domain (spectra) or time domain (signals). The data for each instance is in a different cell.

   b) dwellTime / s

      Time between the acquisitions of any two of the data points.

   c) Samples

      Number of complex data points

   d) ppmAxis / ppm

      Full bandwidth of the acquired spectrum split into number of samples (in ppm)

e) timeAxis / s

0 to number of samples times by dwell time in seconds, split into number of samples.

f) imagingFrequency /MHz

Excitation frequency.

2) instanceNum

The instance to be fitted.

3) voxelNum

The number of the voxel to be fitted.

4) beginTime / s

The time from the excitation to the acquisition of the first FID point. Used for first order phase correction.

5) expOffset / ppm

The expected offset for the reference peak vs. centre of readout in experiment.

6) Pk

A structure containing the prior knowledge. It can be returned by e.g.

```
pk = AMARES.priorKnowledge.PK_SinglePeak
```

7) showPlot

This is either a boolean for whether to show the graphical fit or a figure handle for the fit. The plotting function is relatively slow, and can be turned off for speed when multiple voxels are fitted.

```
[fitResults fitStatus figureHandle CRBResults] =
amaresFit(inputFid, exptParams, pk, showPlot)
```

1) inputFid

Double precision array containing the FIDs to be fitted.

2) exptParams

A structure with the following fields: samples, imagingFrequency, timeAxis, dwellTime, ppmAxis, offset and beginTime.

3) pk

Prior knowledge structure, as for `amares`.

Other options

Offset

The offset is determined in `amares` by finding the maximum of the convolution of the initial value spectrum and the data. If this method gives an offset that is more than 2ppm different from the expected offset, the highest intensity peak is set to 0ppm.

There are several related options that can be passed in to `amares` as a name/value pair:

- 'fixOffset' skips all attempts to find the offset and sets it to a given value (in ppm).
- 'noConvOffset' skips the convolution offset finding and reverts to finding the maximum point within the search limits.
- 'searchlimit' sets the range (in ppm) each side of the expected offset that the peak is allowed to be located in before the maximum point method is used. Default value is 2ppm.

`amaresPlot`

`amaresPlot` is the function that is called to plot the fit of each spectra (see Fig. 1).

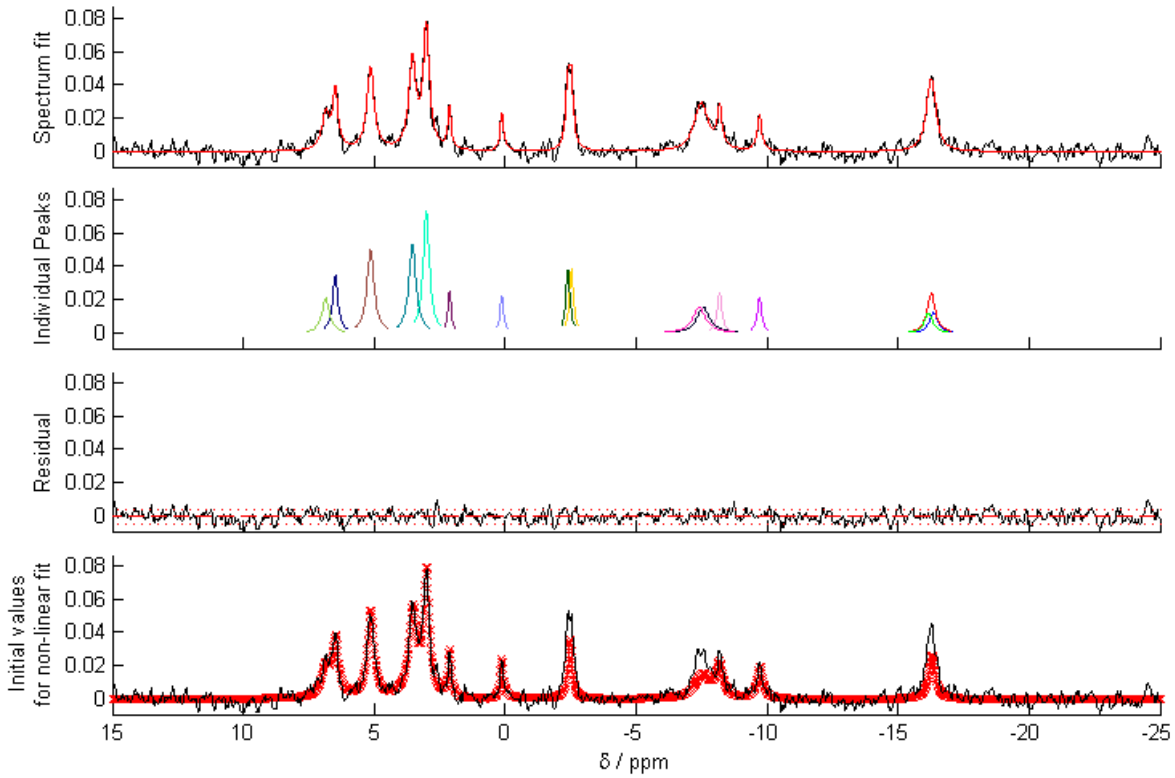

Figure 1: Output of `amaresPlot` for a normal fit of simulated liver ^{31}P data.

By default the figure called by `amaresFit` contains four subplots: the apodized spectrum with overlaid fit, the separate peaks in the fit, the residual, and the initial values. But `amaresPlot` is highly customizable. All inputs are passed in as a name/value pair. Here are a few possible options:

- 'apodization' is the level of apodization on the plotted spectrum. The default value is 30Hz.
- 'xUnits' are the units of the plotted x axis. 'PPM' by default, but can also be set to 'HZ'
- 'firstOrder' applies first order phase correction by default

lsqcurvefit

Options are passed in as `fitOpt`. The two that can be useful to adapt are the maximum number of iterations `fitOpt.MaxIter` (which can be set at the top level with the name/value field name 'MaxIter'), and the tolerance function `fitOpt.TolFun`. `MaxIter` is set to avoid `amares` taking too long to fit spectra of noise. `TolFun` causes the fitting to stop after too few or too many iterations if poorly set. It is scaled with the square root of the max value.

Prior Knowledge

Using prior knowledge

The prior knowledge files are all saved in `AMARES.priorKnowledge` as e.g. `PK_7T_Cardiac.m`. These are assembled into structures when called by e.g. `pk = AMARES.priorKnowledge.PK_SinglePeak`. The structure has three fields: bounds, initial values and prior knowledge. The first two of these are set in the same way as jMRUI. Each peak has a different label. The labels for multiplets are grouped together in a cell. The prior knowledge field contains the relationships between each peak. All the field names within the prior knowledge beginning with 'G_' group peaks together. For example, all peaks with the same number in `G_phase` will have the same phase. There is also the field `refPeak`. This should be assigned to the highest expected peak in the spectrum.

Adding new prior knowledge parameters

The prior knowledge is applied using `applyModelConstraints` and `createModelConstraints`. `createModelConstraints` creates a function for each peaks's linewidth, chemical shift, phase and amplitude. These are set using the prior knowledge. Some small changes might have to be made to the application of initial values in `amaresFit`.

Example using linewidth-constraints

The prior knowledge for linewidth-constrained fitting is `PK_7T_Cardiac_t2`. There are only a few changes between this and `PK_7T_Cardiac` (the normal prior knowledge). The main change is the addition of a "`base_linewidth`" field, which sets the difference between the linewidths of the reference peak and the other peaks. `createModelConstraints` then checks for this field, and if it is found it changes the linewidth function from:

```
linewidth_fun{p+m_count} = {'@(x,a)x(a);',v};
```

to:

```
  linewidth_fun{p+m_count} =
{'@(x,a,b)x(a)+b;',v,pk.priorKnowledge(p).base_linewidth};
```

`applyModelConstraints` has a switch for the function string, which simplifies to:

```
switch thisFunc{1}
        …
         case '@(x,a)x(a);'
             thisOut(idx) = optimVar(thisFunc{2});
         case '@(x,a,b)x(a)+b;'
             thisOut(idx) = optimVar(thisFunc{2})+thisFunc{3};
       …
end
```

`thisFunc` is `linewidth_fun` in this case. `optimVar` is the variable that is being fitted. In this case the base linewidth is a constant that is added to the fitted variable.

Converting between time and frequency domains

The conversion between the two domains can be done by using the functions `specFft` (FID to spectrum), and `specInvFft` (spectrum to FID).

Phasing

The first order phase correction is done by setting the begin time (see above). The zeroth order phase is a fitted parameter based on the reference peak.