OXSA v2 update

The aim of this code update was to simplify the implementation of additional prior knowledge in the AMARES algorithm. This requires minimizing the number of functions that must be changed, and removing redundant code. The major changes are detailed below.

The two parts of the AMARES implementation are the model itself and the prior knowledge.

The model input was consolidated from three function (`makeSyntheticData`, `makeModelFid`, and `makeModelFidAndJacobianReIm`) into `makeModelFid`, which can then be called by various functions based on specific requirements e.g `makeModelFidAndJacobianReIm` for fitting.

If any changes are made to the fitting model, the analytical Jacobians must be determined to pass to `lsqcurvefit`, and to allow calculation of CRLBs. These derivatives were hard coded into various functions, and so were consolidated into `compute_Jacobian`.

If the model is changed, the prior knowledge files must apply to the new model. This is achieved in `createModelConstraints`. This loops through all the compounds and peaks to create functions that describe the relationship between each of the components. There is no option except to hard code in all the possible relationships, which means that this code can be quite confusing. To try and simplify it, indices were relabelled, and additional comments were added.

These functions are then applied in various locations, such as `applyModelConstraints` or `compute_P_Matrix`. In each case, explicit knowledge of the names of each of the variables (e.g. amplitudes or linewidths) is not required, so the code was updated to avoid this. However, the knowledge of how the different parameters are combined into matrices is needed. This is now given in `getCanonicalOrdering`.

In some cases, additional prior knowledge will have an effect on the initial values. This can be either achieved through careful writing of the prior knowledge file, but can also be included as check in `initializePriorKnowledge`, which updates the initial values based on a linear least squares fit of the spectrum.

Another piece of code that affects the prior knowledge is the offset determination in `amares`. Both this and `initializePriorKnowledge` require a model spectrum to fit.

Previously, this was achieved by `makeSyntheticSpectra`. However, this function does not include all the prior knowledge information, and so `makeInitialValuesModelFid` was written to more closely match the final model constraints. This leads to some concrete differences between the updated code and the original, rather than just a refactoring.

To summarise, the steps for updating the fitting model are now:

1. Decide the form of the new model
2. Update the model in `makeModelFid`
3. Calculate analytical derivatives and implement in `compute_Jacobian`
4. Update `createModelConstraints`
5. Update `getCanonicalOrdering`
6. Update Results structure naming in `amares`

And the steps for adding additional prior knowledge are:

1. Include prior knowledge in a PK file
2. Update `createModelConstraints`
3. Update `initializePriorKnowledge`
4. Update `makeInitialValuesModelFid`

Testing

The tests for changing the code should be simple to run and be possible to include in the distribution. Therefore the code for testing was based on the example code and sample data. Monte Carlo simulation are run by adding random noise to two types of high SNR spectra: simulated and phantom data.

For the simulated spectra, noise of twelve variances was added to the perfect example cardiac spectrum. This was repeated 2000 times and each fit was timed. To make this repeatable, the top and bottom 0.1% based on absolute values of each variable, and the top 1% based on absolute CRLBs was excluded, and the remaining data was split into twelve bins of 1143 points based on SNR. The bias, standard deviation, and RMSE were calculated as percentages using the input simulation parameters. The RMSE for the PCr peak is given in Fig 1. The average time to fit each spectrum was reduced by 21% with the restructured code.

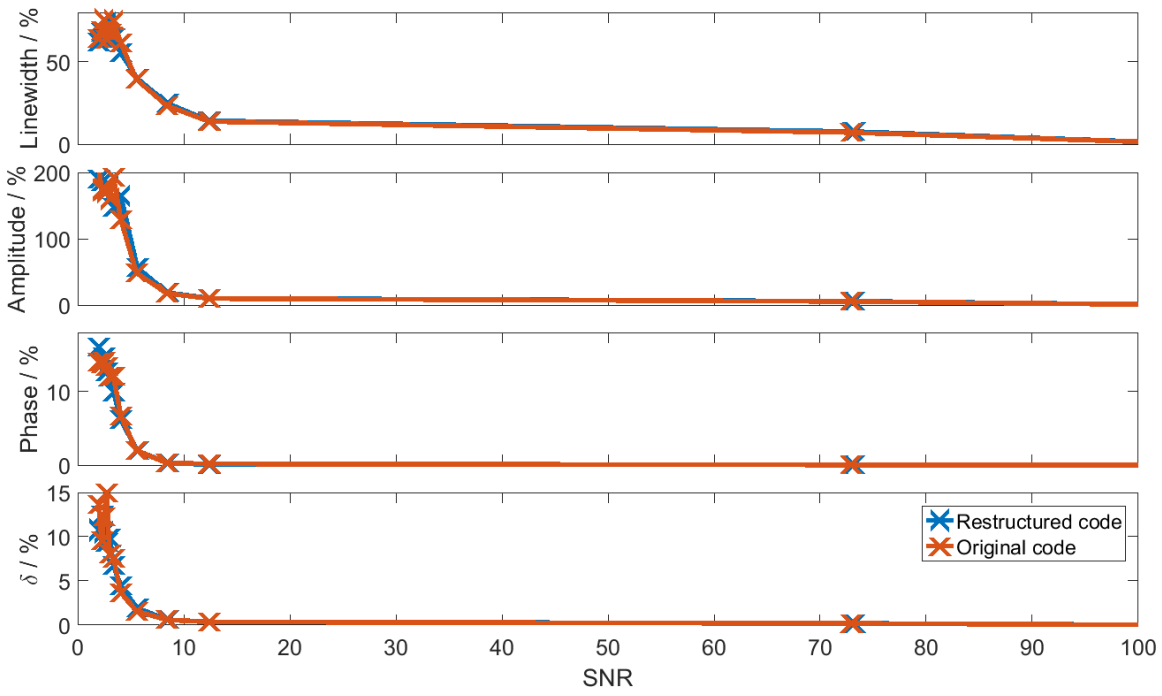

Fig 1. RMSE (%) of PCr fit parameters in Monte Carlo simulation of cardiac spectra. Linewidth, amplitude, phase and chemical shift (δ) RMSE are all given as percentages of the simulation input parameters. The restructured code is in blue, and the original code is given in orange.

For the phantom data, noise of twelve variances was added to the five sample voxels 1000 times each. The extreme values were excluded in the same way as for the simulated data leaving 4764 points per SNR bin. The bias and percentages were calculated from the fit values when no additional noise was added to the spectra. The RMSE is given in Fig 2. Chemical shift is not included, as it is approximately zero except when fitting fails at very low SNR. The average time to fit each spectrum was reduced by 12% with the restructured code.

For both of these tests, the results of the restructured code only differ from the original code at low SNR, where the initial value determination has a greater effect.

This testing code is included with the distribution, as well as reference results to allow the determination of the effect of changes to the fitting algorithm.

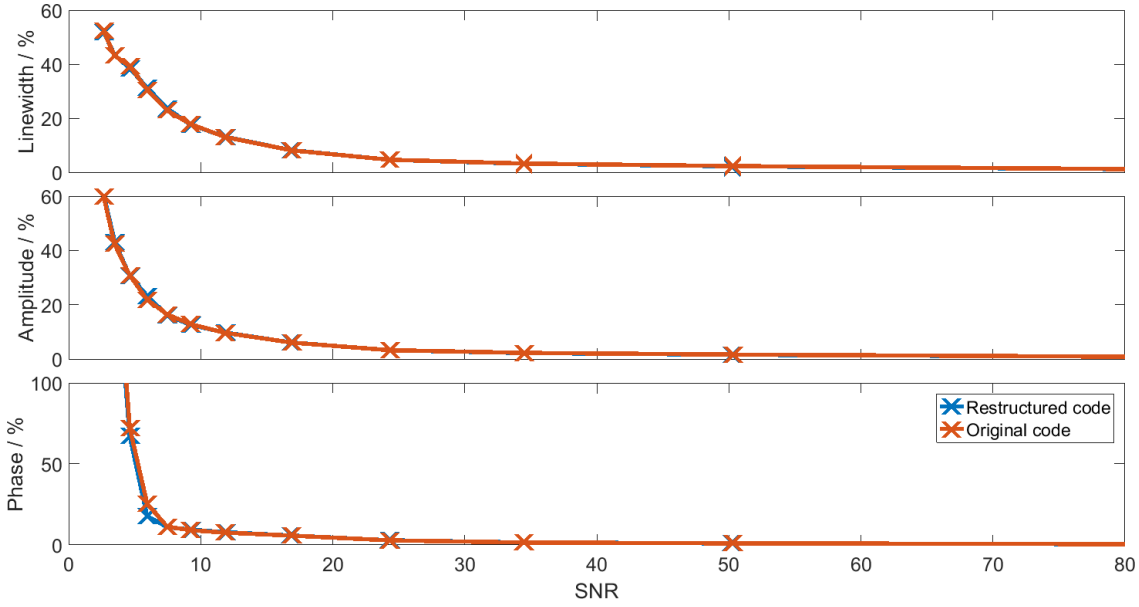

Fig 2. RMSE (%) of fit parameters in Monte Carlo simulation of phantom spectra. Linewidth, amplitude, phase and chemical shift (δ) RMSE are all given as percentages of the fit values when no additional noise is added to each voxel. The restructured code is in blue, and the original code is given in orange.