Coursework 1: Parametric models

This coursework accompanies the first half of GV17/M077: Computational Modelling for Biomedical Imaging. It divides into subsections that roughly reflect the main sections of material in the lectures. After lecture 1, you should be able to complete most of section 1.1.

In each section there is some core material, which is a fairly structured task for you to complete in matlab using the data provided. Each section also contains some further tasks that are more open ended. To get a basic pass (score of 50%) you need to have a decent attempt at the core material; to score more, you will need in addition to have a go at some of the further tasks. I do not expect you to try all of the suggested further tasks, but pick some and do something interesting within them. For a distinction level mark (>70%), I expect to see some innovation within those further tasks: show me an interesting result I have not explicitly asked for.

Your final submission should contain: a short written report (**max 3 sides**), documenting what you have done; figures presenting your results (as many as you like within reason, but they should all be referred to from the report, and each should have a short caption explaining what it is); a code listing, which the report should also explain so that I can find which code corresponds to which experiment or result if necessary. However, don't expect me to look at the code! A well-written report should mean I don't have to and you lose at least a point in the marks every time I feel compelled to check the code. Please make sure all submitted documents include your name.

1.1 Parameter estimation and mapping

This part of the coursework uses a standard high angular resolution diffusion imaging (HARDI) data set acquired from a live human volunteer. It has 3 b=0 images and 30 diffusion weighted images with different gradient directions and a b-value of 1000 s/mm^2. You can download the data and associated files from here: http://cmic.cs.ucl.ac.uk/camino//index.php?n=Tutorials.DTI. Either follow the steps in the tutorial to use Camino to generate dwi.Bfloat or download it directly from the course website: http://moodle.ucl.ac.uk/course/view.php?id=16377.

Once you have the data file, you can load it into matlab as follows:

```
fid=fopen('dwi.Bfloat', 'r', 'b');
dwis = fread(fid, 'float');
fclose(fid);
```

Reshape the data array to reflect the image dimensions: each of the 33 image volumes has 50 slices with 112x112 voxels. The voxel dimensions are 2x2x2 mm^3.

```
dwis = reshape(dwis, 33, 112, 112, 50);
```

Display a single slice to verify we have loaded it in correctly:

```
% Middle slice of the first image volume, which has b=0
imshow(squeeze(dwis(1,:,:,25)), []);
% Middle slice of the second image volume with b=1000
imshow(squeeze(dwis(2,:,:,25)), []);
```

Take the time to understand what the commands above are doing.

You will also need the gradient directions stored in grad_dirs.txt, which we can load into matlab like this:

```
qhat = load('grad_dirs.txt')';
```

and the b-value array, which we can construct like this:

```
bvals = 1000*sum(qhat.*qhat);
```

Now to the real work... The task for this section is to map the parameters of the ball-and-stick model (see lecture slides) over the brain. We will use the ball-and-stick model to measure, indirectly, the density of axon fibres in each voxel and thus produce a map of that parameter over the brain. I will help you get started by showing you code that fits the model, in a fairly naïve way, using non-linear least squares in one single voxel. The rest is up to you.

To start with, let's extract the set of 33 measurements from one single voxel; we'll take the one near the centre of the image:

```
Avox = dwis(:,52,62,25);
```

Before we can run our fitting algorithm we need to implement the ball-and-stick model and a function to minimize (the objective function) for finding the best parameter settings. The following function computes the sum of square differences between the measurements `Avox` and those predicted by the model with parameter settings `x`.

```
function [sumRes, resJ] = BallStickSSD(x, Avox, bvals,
qhat)

% Extract the parameters
S0 = x(1);
diff = x(2);
f = x(3);
theta = x(4);
phi = x(5);
```

```
% Synthesize the signals
fibdir = [cos(phi)*sin(theta) sin(phi)*sin(theta)
cos(theta)];

fibdotgrad = sum(qhat.*repmat(fibdir, [length(qhat)
1])');

S = S0*(f*exp(-bvals*diff.*(fibdotgrad.^2)) + (1-f)*exp(-
bvals*diff));

% Compute the sum of square differences
sumRes = sum((Avox - S').^2);
```

Here is code for fitting the ball and stick model in this voxel using the objective function above with comments to explain each step:

```
% Define a starting point for the non-linear fit
startx = [7.5e+05   3e-03   2.5e-01 0 0];

% Define various options for the non-linear fitting
% algorithm.
h=optimset('MaxFunEvals',20000,...
    'Algorithm','levenberg-marquardt',...
    'TolX',1e-10,...
    'TolFun',1e-10);

% Now run the fitting
[parameter_hat,RESNORM,EXITFLAG,OUTPUT]=fminunc('BallStic
kSSD',startx,h,Avox,bvals,qhat);
```

Run the code above to check that everything is working correctly. After completion of the fitting routine, `fminunc`, I find that the variable `parameter_hat` contains the fitted parameters:

```
parameter_hat =

   7.5000e+05   2.4354e-03  -3.5558e-02   6.5446e-01  -
6.8641e-01
```

in the order S0, d, f, theta, phi. The variable `RESNORM` contains the value of the objective function for the fitted parameters:

```
RESNORM =

   1.2242e+12
```

However, results may vary among versions of matlab and operating system (I used matlabR2014b on a mac).

Familiarize yourself with the list of options available for `fminunc` via the

`h=optimset(…)` command, which the matlab help page for `fminunc` lists. The `'Display'` option is useful. Play around with the options `'MaxFunEvals', 'MaxIter','TolX', 'TolFun'` in particular and observe their effect.

Core questions (1.1.1 to 1.1.4; answer all of these)

Q1.1.1. Think about the quality of the fit you obtain above. First, eyeball the fit to the data by comparing the predictions from the model with your fitted parameters to the actual measurements, as in the lectures. Then, is the final value of RESNORM above the kind of value for the sum of square differences we expect? The standard deviation of the noise on each measurement of this data set is somewhere in the range 5000-6000. Given that, what would you expect a typical value of RESNORM to be? Are the parameter values sensible given the quantities they represent?

Q1.1.2. Adapt the implementation using the transformation method with `fminunc` so that it allows only physically realistic settings of the parameters: S0 must be positive, d must be positive and f must be in [0, 1]. For example, to ensure that S0 is always positive, we can write it as `x(1)^2` rather than `x(1)`. Do we need constraints for theta and phi?

Rerun the fitting above to confirm that the fitted parameters are realistic. Notice that you need to use the inverse transformation to maintain the same starting point. Also `parameter_hat` no longer contains the parameters S0, d, f, theta, phi directly: you need to use the same transformation to get the model parameters. How do the model parameters compare to those from the original implementation? How does the new implementation affect `RESNORM`? Why?

Q1.1.3. We need to assess whether we have really found the global minimum of our objective function and thus the best estimate of the model parameters. To do this, repeat the fitting procedure above lots of times from different starting points. To get each starting point, add normally distributed random numbers to the one we used above; the scale of each random number should reflect the scale of each individual parameter to ensure large enough perturbations.

In what proportion of trials do we find the solution with the smallest value of `RESNORM`? Do you think this is the global minimum? How many runs do we need to be 95% sure we find the global min?

Try some other voxels. How consistent are your answers above?

Q1.1.4. Now create parameter maps over one slice of the image volume, say the middle slice `dwis(:,:,:,25)`, by repeating the procedure above in each voxel of the slice. Produce maps of S0, d, f and the residual error `RESNORM`. Then produce a map of the fibre direction **n** from the parameters theta and phi; Matlab's quiver function might be useful here. To get a more coherent fibre direction map, weight **n** by the volume fraction f before plotting.

Further questions (don't answer all of these)

Q1.1.5. Informed starting points can greatly increase the likelihood of finding the global minimum. We can obtain good guesses for the parameters of the ball and stick model from the diffusion tensor model, which we can estimate from a linear fit (much faster). Implement the linear diffusion tensor estimation (see lecture notes and below) and use it to define a starting point for the ball and stick fitting routine. Repeat steps 1.1.1 to 1.1.4 above and calculate the difference in computation time required to generate the parameter map.

The diffusion tensor model is

$$S(b,\hat{\mathbf{q}}) = S(0,0)\exp\left(-bd\hat{\mathbf{q}}^T D\hat{\mathbf{q}}\right)$$

Taking logs we can write the right hand side like this:

$[1\ -bq_x^2\ -2bq_xq_y\ -2bq_xq_z\ -bq_y^2\ -2bq_yq_z\ -bq_z^2]\ .\ [\log S(0,0)\ D_{xx}\ D_{xy}\ D_{xz}\ D_{yy}\ D_{yz}\ D_{zz}\ ] =$ **y**.**x**.

Thus to estimate the diffusion tensor, we must solve the matrix equation **S** = Y **x**, where **S** is the vector of log measurements, Y is the design matrix with rows as **y** above (one row for each measurement), and **x** is the list of parameters as above.

Try a few potential mappings from diffusion tensor estimate to ball and stick starting point. Which maximizes likelihood of finding the global min?

Q1.1.6. An alternative way to constrain your parameter estimates to realistic ranges, as opposed to the transformation method in Q1.1.2, is to use constrained optimization via the `fmincon` function. Look up the help page and implement the constraints this way. Compare computation times and global minimum location statistics with the results from Q1.1.2.

Q1.1.7. Add the derivative of the objective function with respect to the model parameters to your version of `BallStickSSD.m`. To make sure they are correct, look up the help page for `fminunc` and try out the `'DerivativeCheck'` option. How does the inclusion of the derivatives affect a) the frequency of finding the global minimum, b) the computation time?

Q1.1.8. Above we assume a Gaussian noise model on our measurements by using the sum of squared differences objective function. However, a more accurate model for the noise on MRI voxel intensities is the Rician distribution. Implement an alternative objective function for the Rician noise model. How much does it affect the parameter estimates and computation time?

1.2 Uncertainty estimation

This part continues with the same data as the previous subsection, but extends from estimating a single maximum likelihood estimate to estimating uncertainty and confidence intervals on each parameter.

Core questions (1.2.1 to 1.2.2; answer all of these)

Q1.2.1. Use the parametric bootstrap procedure to estimate the 2-sigma range and 95% range for the parameters S0, d and f in the ball and stick model in the single voxel you used in Q1.1.1. Try some other voxels and compare the estimates.

Q1.2.2. Use MCMC to provide another estimate of the 2-sigma range and the 95% range for the same parameters as in Q1.2.1. How does it compare with the parametric bootstrap?

Further questions (don't answer all of these)

Q1.2.3. Try Laplace's method (look at the help pages of `fminunc` or `fmincon` to see how to return the objective function Hessian) and the various flavours of non-parametric bootstrap discussed in the lectures to get more estimates of parameter confidence intervals. Compare the parameter confidence intervals you estimate with those from Q1.2.1 and Q1.2.2. Comment on any differences you observe.

Q1.2.4. Create a visualization of the uncertainty in the estimate of fibre orientation mapped over a whole brain slice. Can you devise an equivalent for the orientation parameter to the 95% range on the scalar parameters S0, d and f?

Q1.2.5. Implement a simple tractography procedure. Use the various resampling techniques above to study the distribution of tractography trajectories from a single starting point. How consistent is it among estimates of the confidence intervals?

1.3 Model Selection

We'll switch to a different data set for this part. You can download the file `challengeOPEN.txt` from the Moodle website. It is the data set that Uran Ferizi used in his paper "A ranking of diffusion MRI compartment models with in-vivo human brain data" *Magnetic Resonance in Medicine* 2014, which you can also access from the Moodle site, and formed the basis of the MICCAI 2013 White Matter Modelling Challenge. The ISBI 2015 conference is holding a similar challenge this year, although the data set is slightly different; the website gives some additional information: http://biomedicalimaging.org/2015/white-matter-modeling-challenge/.

Here is some code that loads the data and independent variables describing the device settings for each measurement into matlab and computes the gradient directions and b-values, etc, that you'll need to fit models in the tasks that follow.

```
datafile = 'challengeOPEN.txt';
fid = fopen(datafile, 'r', 'b');
```

```
% Read in the header
A = fgetl(fid);

% Read in the data
A = fscanf(fid, '%f', [8, inf]);
fclose(fid);

% Create the protocol
meas = A(1,:)';
grad_dirs = A(2:4,:);
G = A(5,:);
delta = A(6,:);
smalldel = A(7,:);
TE = A(8,:);

GAMMA = 2.675987E8;
bvals = ((GAMMA*smalldel.*G).^2).*(delta-smalldel/3);
```

Note that this data set is not an image, but a set of measurements from a single voxel; the values actually come from averaging over an area of similar voxels. The total number of measurements per voxel is much larger than in the data set we used for sections 1.1 and 1.2: 1152 rather than 33. That allows us to fit some more interesting models.

You may find it useful to work through the Camino tutorial here, http://cmic.cs.ucl.ac.uk/camino//index.php?n=Tutorials.WMAnalyticModels, which uses a similar data set, but that is not essential to continue.

Core questions (1.3.1 to 1.3.3; answer all of these)

Q1.3.1. Adapt your code from Q1.1.3 to fit the ball and stick model to this new data. Bare in mind that the best starting point, constraint encoding, and choice of optimization algorithm may be quite different to those you found for the other data set.

Q1.3.2. Adapt the code further to fit various different models listed below as well as the diffusion tensor model. Each model, of course, will need a different encoding to impose appropriate constraints, different algorithms to determine a suitable starting point, and will require different numbers of random starting points to ensure finding the global minimum fit.

Two compartment models all have the general form

$$S(b,\hat{\mathbf{q}}) = S(0,0)\left(fS_I(b,\hat{\mathbf{q}}) + (1-f)S_E(b,\hat{\mathbf{q}})\right),$$

where S_I is the signal from water inside cells (intra-cellular signal) and S_E is the signal from water outside cells (extra-cellular signal). For now, we'll only consider one model for the intra-cellular signal, which is the stick model where

$$S_I(b,\hat{\mathbf{q}}) = \exp\left(-bd(\hat{\mathbf{q}}.\mathbf{n})^2\right),$$

d is the diffusivity and the unit vector **n** is the fibre orientation.

The new models below have models for S_E that differ from the ball and stick model.

Ball and stick model revisited: in the ball and stick model, the extra-cellular signal uses the ball model:

$$S_E(b,\hat{\mathbf{q}}) = \exp(-bd)$$

in which the signal is independent of orientation.

The *zeppelin and stick model* uses the zeppelin model of the extra-cellular signal. The zeppelin model is a diffusion tensor with rotational symmetry around the principal axis, ie it has two equal eigenvalues. We will assume for now that the two smaller eigenvalues are equal and the largest eigenvalue is associated with the fibre direction **n**. Thus,

$$S_E(b,\hat{\mathbf{q}}) = \exp\left(-b(\lambda_2 + (\lambda_1 - \lambda_2)(\hat{\mathbf{q}}.\mathbf{n})^2)\right)$$

where $\lambda_1 >= \lambda_2 > 0$ are the two unique eigenvalues.

The *zeppelin and stick with tortuosity model* is as the zeppelin and stick model above, but assumes that $\lambda_2 = (1-f)\lambda_1$.

Q1.3.3. Use the AIC and BIC to rank the three models. The standard deviation of the signal in this data set is about 0.04. Are the rankings from each criterion consistent? What do you conclude from the ranking(s)?

Further questions (don't answer all of these)

Q1.3.4. Repeat the model selection experiment above, but evaluating the posterior probability via Bayesian model selection to rank the models. How does it compare to the more ad-hoc criteria used in Q1.3.3?

Q1.3.5. Extend the range of models to include some more exotic beasts. The set of models in Panagiotaki et al NeuroImage 2012 certainly contains some that should do better than any of those listed in Q1.3.2. Replacing the stick model with a cylinder is one option, although coding the cylinder model is a little fiddly. Another possibility is to include multiple stick compartments to relax the assumption that all the axon fibres in the voxel that produced the data are perfectly parallel. For example the ball-and-two-sticks model is

$$S(b,\hat{\mathbf{q}}) = S(0,0)\left(f_1 S_I(b,\hat{\mathbf{q}};\mathbf{n}_1) + f_2 S_I(b,\hat{\mathbf{q}};\mathbf{n}_2) + (1-f_1-f_2)S_E(b,\hat{\mathbf{q}})\right)$$

where S_E is the ball model and each S_I term has its own fibre orientation. See also the fibre dispersion models in (Zhang et al NeuroImage 2012), (Sotiropoulos et al NeuroImage 2012), and (Ferizi et al MICCAI 2013).

Prize question! Small prize awarded for the person that can construct the model with the smallest AIC on this data. To be eligible, you must do better than my best attempt, which is the ball-and-two-sticks model above. To enter, send me your code for fitting the model, the best set of parameter that you have found, and what sum of square differences you get; I'll compute the AIC myself and compare it with that for my implementations of all the models I've discussed. Don't feel you have to be limited to compartment models; any general kind of regression model might also do the job.

Q1.3.6. Try fitting the simple two compartment X-and-stick models in Q1.3.2 to the imaging data we used in sections 1.1 and 1.2. Use whichever model selection criterion you prefer to select the most appropriate model in each voxel of the image and create a map indicating which model is most appropriate at each location. What do you observe? Does the map show consistency between the left and right sides, which would suggest some anatomical meaning? Can you explain why it selects certain models in certain regions? Does it help to average parameter estimates from different models using the Akaike weights?

1.4 Experiment Design

This part uses with the same data as subsection 1.3 to illustrate the basic principles of experiment design.

Further questions (don't answer all of these)

Q1.4.1. Write a function to compute the Fisher information matrix for the ball and stick model. Evaluate the function at the maximum likelihood estimate of the ball and stick parameters that you obtained in **Q1.3.1** for the imaging settings in `challengeOPEN.txt`.

Q1.4.2. The imaging protocol divides into 24 "shells" of 48 consecutive measurements. Each shell contains 3 measurements with b=0 and 45 measurements with equal b>0 but unique gradient orientation. Evaluate the A-optimality criterion (trace of the inverse Fisher information matrix) for each individual shell to determine which single shell provides the best economical protocol for estimating ball-and-stick parameters. I suggest you ignore the orientation parameters, which simplifies evaluation of the A-optimality, although you can incorporate them if you are careful.

Repeat the evaluation but now accounting for the loss of signal from T2 decay that differs among shells, which is an exponential function of the echo time TE:

$$S_0 = M_0 \exp\left(-\frac{TE}{T_2}\right).$$

How does accounting for T2 decay affect the optimal choice of shell?

Does parametric bootstrap of the ball-and-stick model support the choice of shell that A-optimality suggests?

Q1.4.3. Extend **Q1.4.2** to determine the advantage of using a pair of shells with different b-value rather than just one shell. For fixed number of measurements, does a combination of non-zero b-values offer any advantage?