Computational Modelling for Biomedical Imaging - Coursework 1

Razvan Valentin Marinescu
Student Number: 14060166
razvan.marinescu.14@ucl.ac.uk

February 9, 2015

Q1.1.1

From eyeballing the data in fig. 1 we notice that the fit is very poor, especially for the 3 entries where the b-value is 0, having a $\Delta S > 7 * 10^5$. The other entries don't fit the data well either.

The final value of RESNORM (1.2242e+12) is clearly above the kind of value we would normally expect. The fit is poor probably because the search strategy got stuck in a local minima.

The expected value of RESNORM would be given by the formula:

$$E\left[\sum_{i=1}^{33}(X_i - \bar{X}_i)^2\right] = E\left[\sum_{i=1}^{33}\sigma_i^2\right] = 33 * \sigma^2$$

Since sigma is between 5000-6000 this means that $8.25 * 10^8 < RESNORM < 1.188 * 10^9$.

Some of the parameter values we obtain are not sensible. For example, the value of f is negative, meaning that we have negative intra-axonal diffusion, which is not physically possible. The value of S0 is also really high, when in fact it should be around $1.1 * 10^5$.

Q1.1.2

I did the following transformations:

- $S0 \to S0^2$
- $d \to d^2$
- $f \to (1 + e^{-f})^{-1}$ (sigmoid)

This time the algorithm converges to the following parameters: $S0 = 1.132129e + 05, d = 1.534e - 03 f = 0.575, \theta = -1.03, \phi = -0.11$ with a RESNORM of 1.33e+09. These parameter values are realistic and give us a RESNORM that is around 1000 times smaller. Note that we this time the starting parameters were changed to more realistic values ($S0 = 1.5e + 05, d = 3e - 03 f = 0.5, \theta = 0, \phi = 0$). The fit is plotted in figure 2. The reason we get better parameters and RESNORM is because we introduced the transformations that keep the parameters within reasonable limits and because we used a better starting position.

Q1.1.3

Running the same procedure for gaussian-distributed starting points and for three different pixels gives us the following parameters:

Voxel	S0	d	f	θ	ϕ	GlobalMinCounter(out of 100)
(52,62,25)	1.132e+05	1.534e+03	0.575	2.10	6.17	89
(63,40,18)	1.121e+05	1.829e+03	0.5125	1.19	5.43	83
(50,64,23)	1.214e-05	7.208e-04	0.1955	5.05	5.02	87

GlobalMinCounter represents the number of times the global minimum was found was found. We are quite confident that those are the global minimums, since we made sure that the covariance of the gaussian distribution from which we sample the starting position is large enough to cover a wide area of the space. If we consider the probability p of finding the global minimum in one run, then after k runs the probability of **not** having found the global minimum would be $(1-p)^k$ where k is the number of runs. Then, the minimum number of runs in order to be 95% confident that we will find the global minimum is $\arg\min_k 0.05 > (1-p)^k$. Since $p \approx 0.86$ then $k = 2$, meaning that 2 runs are enough to be 95% confident.

Q1.1.4

The maps for $S0$, d, f, $RESNORM$ and the fibre direction n are shown in figures 3a, 3b, 4a, 4b and 5 respectively. See the comments in the captions for interpretations.

Q1.1.5

I implemented the DTI model and used the following mappings to get to the parameters of the BallStick model:

1. $d = mean(D)$, $f = 0.5(|\lambda'_1 - \lambda'_2| + |\lambda'_1 - \lambda'_3| + |\lambda'_2 - \lambda'_3|)$ where $\lambda'_i = \lambda_i/(\lambda_1 + \lambda_2 + \lambda_3)$ for $i \in 1..3$
2. $d = tr(D)/3$, f as above

3. $d = max(D)$, $f = 0.5((\lambda'_1 - \lambda'_2)^2 + (\lambda'_1 - \lambda'_3)^2 + (\lambda'_2 - \lambda'_3)^2)$, for λ'_i as above

The most efficient was method 1, for which the global minimum was found in approximatively 93/100 runs. The parameters I got using method 1 are:$S0 = 1.13184e + 05$, $Dxx = 0.00125$, $Dxy = -0.00013$, $Dxz = -0.00053$, $Dyy = 0.00043$, $Dyz = 0.00011$ and $Dzz = 0.000782$ with a $RESNORM = 1.4458e{+}09$.

Q1.1.6

The computation time for `fmincon` is around 9x slower when compared to `fminunc` (averaged over 100 trials, `fmincon`=6.68sec, `fminunc`=56.28sec). Nevertheless, the global minimum is almost always found by both functions (87/100 runs). The parameters and $RESNORM$ it converges to are the same as in Q1.1.2.

Q1.1.7

I supplied the gradient to `fmincon`. For 10 runs using gaussian distributed starting positions, the time it takes without gradient is 11 seconds while this gets reduced to 7 seconds if the gradient is used. The conclusion is that supplying the gradient speeds up the computation.

Q1.1.8

I implemented the Rician noise model but I didn't manage to make it work.

Q1.2.1

The histogram of $p(x|A)$ along with the 2σ and 95% ranges are shown for parameters $S0$, d and f in figures 6, 7 and 8 respectively. These have been computed for the voxel (52,62,25). Parameter estimates for various other voxels are given in table 1. A better visualisation[1] of the parameter intervals for $S0$, d and f is given in figures 9, 10, 11. The 2σ and 95% confidence ranges all agree with each other and have roughly the same mean. In table 1 all voxels have similar ranges for $S0$, however voxel (70,64,14) has different ranges for d and f, probably because it is a CSF or gray matter voxel. In figures 9, 10, 11 we notice that the 2σ range is always smaller than the 95% confidence interval, for all parameters.

Q1.2.2

The 2σ and 95% confidence ranges for MCMC are given in figures 9, 10, 11. Ranges reported by MCMC have the same mean as the ranges generated by the parametric bootstrap, but they are larger, suggesting that MCMC covers a larger parameter space.

Q1.2.3

I only managed to implement Laplace's method, which provided smaller parameter ranges than the quivalent ones for parametric bootstrap or MCMC, but all having the same mean. See figures 9, 10, 11.

Q1.3.1

The parameters that I found are: $S0 = 0.9231$, $d = 1.1355e - 09$, $f = 0.49$, $\theta = 4.704$, $\phi = 0.038$ with a $RESNORM = 3.8681$. The predicted signal is plotted against the measurements in figure 12. The parameters have been found by applying `fmincon` for several batches of 100 runs, with random starting positions sampled from a gaussian distribution with mean μ. Parameter vector μ was updated at each iteration with the global minimum found in the previous round.

Q1.3.2

For fitting these models I decided to switch to `fminunc` because it is faster and slightly more efficient at finding the global minimum. For the Zeppelin-Stick model I actually made implementations with both functions and realised that for the same starting position, `fminunc` was converging to better parameters and $RESNORM$ than `fmincon`. The fit for these models can be visualised in figures 13, 14 and 15. For finding the best fit the same protocol was used as in Q1.3.1. The best model was the Zeppelin-Stick, with a $RESNORM = 1.1784$, followed by Zeppelin-Stick with tortuosity, Ball-stick and DTI having $RESNORMS$ of 2.1337, 3.8681 and 20.5832 respectively. The main reason the Zeppelin-Stick model outperforms the others is because it is more general than ZeppelinStick with tortuosity and Ball-Stick, having more parameters. It is therefore expected to perform at least as good as the others.

[1] plot contains two other methods apart from parametric bootstrap: MCMC and Laplace

Q1.3.3

	Zeppelin-Stick	Zeppelin-Stick Tort	Ball-Stick	Diffusion Tensor
AIC	**750**	1345	2427	12878
BIC	**785**	1375	2452	12913

The AIC and BIC scores show that the Zeppelin-Stick model is the one performing best (having the lowest scores), followed by Zeppelin-Stick with tortuosity, Ball-Stick and DTI. AIC and BIC rankings are consistent across the models and actually yield very similar scores.

Q1.3.5

I implemented three extra models: Ball-Two-Sticks, Zeppelin-Two-Sticks, Ball-Watson and Zeppelin-Watson. The latter two models are based on a simplified version of the NODDI model, where the intra-cellular component is modelled using a Watson distribution[2], while the extra-cellular component is modelled using a ball or zeppelin respectively. The Zeppelin-Watson model is defined as follows:

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

$$S_I(b, \hat{\mathbf{q}}) = \int_{\mathbb{S}^2} f(\mathbf{n}) e^{-bd(\hat{\mathbf{q}}n)^2} d\mathbf{n}$$

$$f(\mathbf{n}) = M(0.5, 1.5, k)^{-1} e^{k(\boldsymbol{\mu}\mathbf{n})^2}$$

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

The Ball-Watson model is done in a similar manner, with the isotropic extra-cellular diffusion instead. Computing the exact analytical result for S_I would have taken too much time, so I approximated the interal using a simple Monte-Carlo method, taking the mean of 20,000 samples on a sphere. The code was initially very slow, but I managed to speed it up with vectorisation. The Ball-Watson model outperformed the Ball-Stick model, but the Zeppelin-Watson model did not beat the Zeppelin-Stick one, although it came very close. The are several combined reasons for this: (1) sampling the integral of S_I took very long (0.6 sec), so I could only manually run `fminunc` a few times with different starting positions, (2) since the computation of the SSD was slightly non-deterministic[3],`fminunc` had problems converging, (3) the fact that I didnd't properly model the extra-cellular space like in the full NODDI.

	Zeppelin-Stick	Ball-Two-Sticks	**Zeppelin-Two-Sticks**	Ball-Watson	Zeppelin-Watson
SSD	1.17	2.96	**1.13**	3.34	1.23
AIC	750	1867	**723**	2105	786
BIC	785	1908	**764**	2135	826

We therefore conclude that Zeppelin-Two-Sticks is the best with an AIC score of 723.

Q1.4.1

The Fisher information matrix evaluated at the optimal parameters is:

2.37087e+05	-4.666e+22	2.49496e+05
-4.666e+22	1.671e+40	-5.913e+22
2.49496e+05	-5.913e+22	5.04470e+05

Q1.4.2

I evaluated the Fisher matrices for each shell and ran both A- and T-optimality. For A-optimality I get shell 22 best with a score of 5.5171e-04, while for T-optimality I get shell 19 best with a score of 1.2164e+39. I didn't manage to account for T2 decay.

[2] I implemented everything myself apart from the sphere sampling and Kummer's (confluent hypergeometric) function M(a, b, z)

[3] as I numberically approximated the integral in S_I