COMP0118 - Coursework1

Francesco Seracini

February 2025

1 Parametric models

1.1 Parameter estimation and mapping

1.1.1

I began by implementing the linear diffusion tensor estimation for one voxel, applying a weighted least squares fit, which has as objective function the minimization of the sum of square differences. I then proceeded to extend this approach to all the voxels in the $72nd$ slice. All the mean diffusivity values were stored inside the dt_map72 and used to create the fractional anisotropy map and the colour-coded principal direction map. The fractional anisotropy map was calculated using the formula: $FA = \sqrt{\frac{3\sum(\lambda_i - \mathrm{Tr}(D)/3)^2}{2\sum\lambda_i^2}}$ where D is the diffusion tensor matrix and λ_i are the eigenvalues of the matrix. To create the last map I extracted the principal eigenvector of D and associated the results with the RGB vectors proportional to the squared fractional anisotropy, as suggested in *"An introduction to computational diffusion MRI: the diffusion tensor and beyond"* by Daniel C. Alexander. The generated maps can be seen in *Figure 1.* Overall, we obtained acceptable results, even though, due to the simplicity of this linear model, significant problems appear along the borders of the brain, where the images are irregular and present an high concentration of black pixels.

1.1.2

In this section I analyzed the results obtained using the provided naive version of the *Ball-and-Stick* model. The objective function is the sum of square differences and it is implemented in *BallStickSSD.m.* The model is characterized by 5 parameters: $S0$, the baseline signal without diffusion weighting, d, the diffusivity coefficient, f, the volume fraction of the stick compartment, θ the elevation angle and ϕ, the azimuth angle. *Figure 2* shows the model's fit to the data by comparing the predicted values with the actual measurements. It is evident that the model does not perform well, particularly for measurements with b values equal to zero. In fact, by estimating the model parameters for the voxel $Avox = dwis(:, 92, 65, 72)$, we obtain the following results: $parameters = [3.5239 \times 10^3 \quad -4.8868 \times 10^{-6} \quad 1.2256 \times 10^2 \quad 8.8104 \times 10^{-1} \quad 1.5531]$ and $RESNORM = 2.8740 \times 10^7$, that show two evident issues. First of all, we can notice that the d and f values are out of their acceptable ranges. The first one must be positive, while the other should be within the range $[0, 1]$. Secondly, the $RESNORM$ value is excessively high. Considering that the standard deviation of the noise on each measurement of this data set is around 200 and that the $RESNORM$ should follow the relation: $RESNORM \leq N\sigma^2$, where $N = 108$ is the number of measurements, the expected upper bound for $RESNORM$ should be: 4.32×10^6. These findings indicate the need for model improvements to achieve better results.

1.1.3

In this section I adapt the implementation in 1.1.2 to allow only physically realistic settings of the parameters: $S0$ must be positive, d must be positive, and f must be in $[0, 1]$, while there are no constraints on theta and phi because they are periodic angles. To achieve this, I squared $S0$ and d, and applied the exponential function to the negative of $f^2(BallStickSSD\_constraints)$. To maintain the same starting point, I performed the inverse transformation before running the optimization process, and then I applied the same transformation to the resulting model parameters. The final results are: $parameters = [4.2579 \times 10^3 \quad 1.1413 \times 10^{-3} \quad 3.5731 \times 10^{-1} \quad -9.8107 \times 10^{-1} \quad 5.7945e \times 10^{-1}]$ and $RESNORM = 5.8720 \times 10^6$. As expected, now all the parameters are in a realistic range, but, even though the $RESNORM$ value is significantly lower than the one obtained in 1.1.2, it is still higher than the expected upper bound. This discrepancy is also visible in *Figure 3*, which shows a better fit compared to *Figure 2*, though it remains far from perfect. The improved results obtained by constraining the parameters are likely due to limiting the search space, thereby reducing the risk of getting stuck in local minima that occur only with unrealistic parameter sets.

1.1.4

To ensure that the global minimum of the objective function was found, I ran the optimization process 1000 times with different starting points. In $find\_optimal\_parameters.m$, I added normally distributed random numbers to each starting values used in the previous section, depending on their order of magnitude, as shown in *Table 1.* At the end of each optimization process I checked if the $RESNORM$ value obtained was lower than the minimum $RESNORM$ found so far. In $min\_resnorm\_percentage.m$, I calculated the proportion of trials where the smallest $RESNORM$ value was found and estimated the required number of runs to achieve a 95% probability of finding the global minimum at least once, applying first the Complement Rule in Probability: $P(\neg A) = 1 - P(A)$, where in this case A is to find no global minimum in N runs, and then the Binomial Probability Theorem: $P(\neg A) = \binom{N}{0} p^0 (1-p)^N$. I found the minimum $RESNORM$ value in 96.4% of the trials, meaning that the required number of runs is 9.0118×10^{-1}, namely 1. I ran the same process for 9 other voxels, and the results are displayed in *Table 2.*

1.1.5

In this section, I applied the same process used in 1.1.4 to all voxels in the 72*nd* slice. In the previous question, I determined that the maximum number of runs required was 10 ($Avox = dwis(:, 77, 95, 72)$), but, considering that I have not tested every single voxel, I will use 15 different starting points to ensure to reach the global minimum. At the end of the optimization process for each voxel, I stored the parameter values and $RESNORM$ in their respective maps. The fiber map is computed by converting the theta and ϕ maps into $x - y$ coordinates and weighting the results by the volume fraction f. The plots of these maps are shown in *Figure 4*. Overall, the figures provide a good visualization of the fit quality as well as the structure, diffusivity, and orientation of the fibers. However, we can observe several numerical errors like black pixels, particularly in the f map, and irregular patterns along the brain's borders.

1.1.6

In this section I analyzed different ways to find the global minimum faster and in a more reliable way. First of all I used the optimization function *fmincon*, the options are displayed in *Table 3*, in which you can specify the constraints for the parameters. The computational time for each iteration was higher but, as we can see from *Table 4*, the maximum number of runs needed to be 95% sure to find the global minimum is 6, while it was 10 using *fminunc*. An interesting thing to notice is that at the same time we never have a number of required runs smaller than one, like it happened for *fminunc*. My opinion is that *fmincon* is more reliable but for bigger or more particular perturbations of the starting parameters it get stuck in a local minimum without the possibility to doing anything. In *Figure 5* it's possible to see that the maps obtained with *fmincon* present less numerical errors and black pixels. I also tried to start the optimization process from the set of parameters obtained from the *Linear Diffusion Tensor* model ($DT\_starting\_point.m$). Like before, the computational time needed for a single iteration became higher, but, unlike with *fmincon*, this time the results of the maps (Figure 6) are worse and the minimum number of runs is greater. I think that is probably due to the size of the perturbations assigned to the $S0$ parameter. In fact the starting point used in the previous question was 4 order of magnitude less than the final value, so a noise STD of 10^3 was the right choice. In this case, starting from a value only slightly different than the optimal one, this perturbations could impede the achievement of the global minimum.

1.1.7

I modified the objective function for the Ball-and-Stick model to account for the existence of the Rician noise ($BallStickSSD\_Rician.m$). I used the simple approximations founded in *"Alexander 2009"*. I proceeded to use it to estimate the parameters of the model, but obtaining disappoint results. In fact it was more time consuming and had as a result a greater $RESNORM$. This is probably due to the nature of the data I'm working with or to the extremely simplified version of the Rician noise model I'm considering. For this reason, I decided that further investigation with this objective function was not necessary.

1.2 Uncertainty estimation

1.2.1

In this section, I aimed to estimate the uncertainty and confidence intervals for each parameter using different procedures. Here, I applied the *Classic Bootstrap* algorithm to estimate the $2-\sigma$ range and 95% confidence interval for the parameters $S0$, d, and f in the *Ball-and-Stick* model for the single voxel used in 1.1.2. I generated 1000 samples with replacement from the original measurements and then ran the same optimization process used in 1.1.4, saving the results in $total\_samples$ (*Figure 7*). Then, using $Two\_sigma\_range.m$ and $NinetyFive\_range.m$, I calculated the two requested ranges. The results are shown in *Table 5*. I performed the same process for four additional voxels, and the results are displayed in *Table 6*. The results obtained from different voxels provide reasonable estimates; however, in some voxels, the confidence ranges are much wider than in others, likely due to the model performing worse in those areas.

1.2.2

I implemented the *Metropolis-Hastings* algorithm, a *Markov Chain Monte Carlo* ($MCMC$) method, using the specifications shown in *Table 7* to obtain another estimate of the $2-\sigma$ range and 95% confidence interval for $S0$, d, and f. First, I ran $find\_optimal\_parameters.m$ to determine a better starting point and computed the squared noise standard deviation using the formula: $\tilde{\sigma}^2 = \frac{1}{K-N}\sum_{k=1}^{K} R_k^2$. Then, in each iteration, a new sample is generated by drawing a random set of parameters from a proposal distribution, which has as its mean the corresponding value of the previous sample and as its standard deviation a value proportional to its dimension. At this point, I calculated the acceptance probability in $acceptancy\_probability.m$ using the formula: $\alpha = min(1, \frac{\sin(\theta_c)}{\sin(\theta_{t-1})} e^{-\frac{2\sigma^2}{1}[SSD(y)-SSD(x)]})$ where I added the sine terms because of the non-uniformity of the distribution along different directions. If the obtained value is higher than a probability randomly drawn from a uniform distribution, the new sample is accepted. The histograms of the sampled values for $S0$, d, and f are displayed in *Figure 8* and the ranges are in *Table 8*
These results closely match those obtained using the *Classical Bootstrap* method, reinforcing confidence in the reliability of our approach.

1.2.3

I calculated additional estimates of the $2-\sigma$ range and 95% confidence interval for $S0$, d, and f using, *Parametric Bootstrap*, *Residual Bootstrap* and *Wild Bootstrap*. The corresponding posterior distributions are displayed in *Figures 9*, *10* and *11* while the different ranges are in *Tables 9*, *10* and *11*. I also implemented the *Laplace Method* to have another estimate of the $2-\sigma$ ranges (*Table 12*. The ranges obtained are all very similar even though the posterior distributions are quite different in shape . I just want to point out that the *Residual Bootstrap*'s $S0$ range has a lower bound significantly lower compared to those obtained from the other algorithms.

1.3 Model Selection

1.3.1

In this section I adapted the code from section 1.1 to fit the *Ball-and-Stick* model to the new data set. I decided to use the same starting point, the same options for *fminunc* and the same transformations for $S0$, f and d. Of course, considering the different intensity of the signals measured, I changed the order of magnitude of the perturbations applied to the starting set of parameters to run the optimization process from different starting points (the perturbation values are shown in *Table 13*).The results obtained are: $parameters = [1.0099 \quad 1.4321 \times 10^{-3} \quad 5.7493 \times 10^{-1} \quad 1.5447 \quad 3.0587]$ and $RESNORM = 1.5106 \times 10^{1}$. And running the process from 1000 different starting points I obtain : $percentage = 9.9900 \times 10^{-1} \quad num\_runs = 4.3368 \times 10^{-1}$. I used the same formula of 1.2 to calculate the expected upper bound for $RESNORM$ and, considering the standard deviation for the noise in this data set is 0.04 and the number of measurements is 3612, we obtain $max\_RESNORM = 5.7792$, which is far lower than the minimum we found. This means that the model is not performing well enough, as we can also see from *Figure 12*.

1.3.2

I adapted the code to fit two new models: *Zeppelin-and-Stick* ($ZeppelinStickSSD.m$) and *Zeppelin-and-Stick with Tortuosity* ($ZeppelinStickTortuosity.m$). The first model has six parameters instead of five, as it requires estimating the smaller eigenvalues, while the largest eigenvalue is equal to the diffusivity coefficient d. For the sixth parameter, I chose not to use the eigenvalue directly but instead defined a parameter s, which I constrained within the range $[0, 1]$, similar to f. This ensures that multiplying s by d results in a smaller eigenvalue. On the other hand, the Tortuosity model has only five parameters because the smaller eigenvalue is defined as $(1 - f) \times \lambda_i$, where λ_i is the largest eigenvalue. I ran the optimization process for these two models as well as for the linear diffusion tensor model, and the corresponding $RESNORM$ values are: $SSD\_DT = 326.98 \quad RESNORM\_ZS = 10.8167 \quad RESNORM\_ZST = 11.6052$. The fit to the data is shown in *Figures 13*, *14*, and *15*. For all models, I also calculated the required number of random starting points to ensure convergence to the global minimum (*Table 14*).

1.3.3

I computed the *AIC* and the *BIC* values for the four models and I ranked them (*Table 15*): As we can see the results from the two criterion shows the same output: the *Zeppelin-and-Stick* model is the best one while the *Linear Diffusion Tensor* is the worst one.

1.3.5

I repeated the steps done in 1.3.1, 1.3.2 and 1.3.3 for the other 5 voxels, calculating all the *AIC* end *BIC* values and reporting them in *Table 16*. We can see that the results obtained are consistent across all voxels.Furthermore, I implemented the *100-Fold Cross-Validation* method, the *Monte-Carlo Cross-Validation* method and the *Leave-one-out Cross-Validation* method. I applied all these methods to the *Ball-and-Stick* model, the *Zeppelin-and-Stick* model and the *Zeppelin-and-Stick with Tortuosity* model to compare the average errors. The results can be seen in *Table 17*, and they validate the superior performance of the *Zeppelin-and-Stick* model compared to the other two models

1.3.6

I computed the *AIC* values for the *Zeppelin-and-Stick* and the *Zeppelin-and-Stick with Tortuosity* models for all the voxel in the $72nd$ slice of the data set used in sections 1.1 and 1.2 and then I created a map, red for *ZST* model and blue for *ZS* model, selecting, for each voxel, the best model. As we can see from Figure 16 the model with *Tortuosity* has the lowest AIC value in the majority of the voxels, but we can observe a certain symmetry in the disposition of the blue dots. In fact, I think that due to the inaccuracy of both models (their $RESNORM$ is higher than the expect lower bound) the goodness of the two models it's almost the same and so the one with *Tortuosity*, that has 5 parameters instead of 6, is chosen because of *Occam's Razor* principle. However, the blue areas, likely at white-gray matter transitions, have less restricted diffusion, making the more flexible *Zeppelin-and-Stick* model a better choice. I then proceeded to compute the Akaike weights and use them to weight the estimated signals obtained from the 2 models and the *Ball-and-Stick* model. The $\log(RESNORM)$ map is shown in Figure 17 and, even though it gives us a better result than the previous ones, it's still far from perfect and it would probably need the development of some anlaytical improvements.

1.4 Experiment Design

1.4.1

In this section I wrote a function to compute the Fisher Information Matrix ($Fisher\_Information\_Matrix.m$) paying attention to choose units to equalize numerical scale and to normalize the matrix. I evaluated it at the maximum likelihood estimate of the Ball-and-Stick parameters that I obtained in 1.3.1 and I obtained:

$$\text{FIM} = \begin{bmatrix} 1.4742 \times 10^{4} & -1.2273 \times 10^{3} & 1.6303 \times 10^{3} \\ -1.2273 \times 10^{3} & 1.3543 \times 10^{4} & -1.0032 \times 10^{4} \\ 1.6303 \times 10^{3} & -1.0032 \times 10^{4} & 1.3325 \times 10^{4} \end{bmatrix}$$

After having divided the protocol into 36 shells, I used the function to compute the *A-optimality* and *T-optimality* value for each shell and I found out that for both criteria the most economical protocol is the 34th shell, the one with $b\_value = 3.021902205 \times 10^{4}$.