**Q1.1.1**

I am not very clear about your question. Mean square error (or sum of squared error) is related to variance and bias as

Given that there is no bias, MSE is linearly related to variance (thus standard deviation). The task of any fitting is to find the mean from the noisy data. If noise is high, the uncertainties in the fitted parameters are also high.

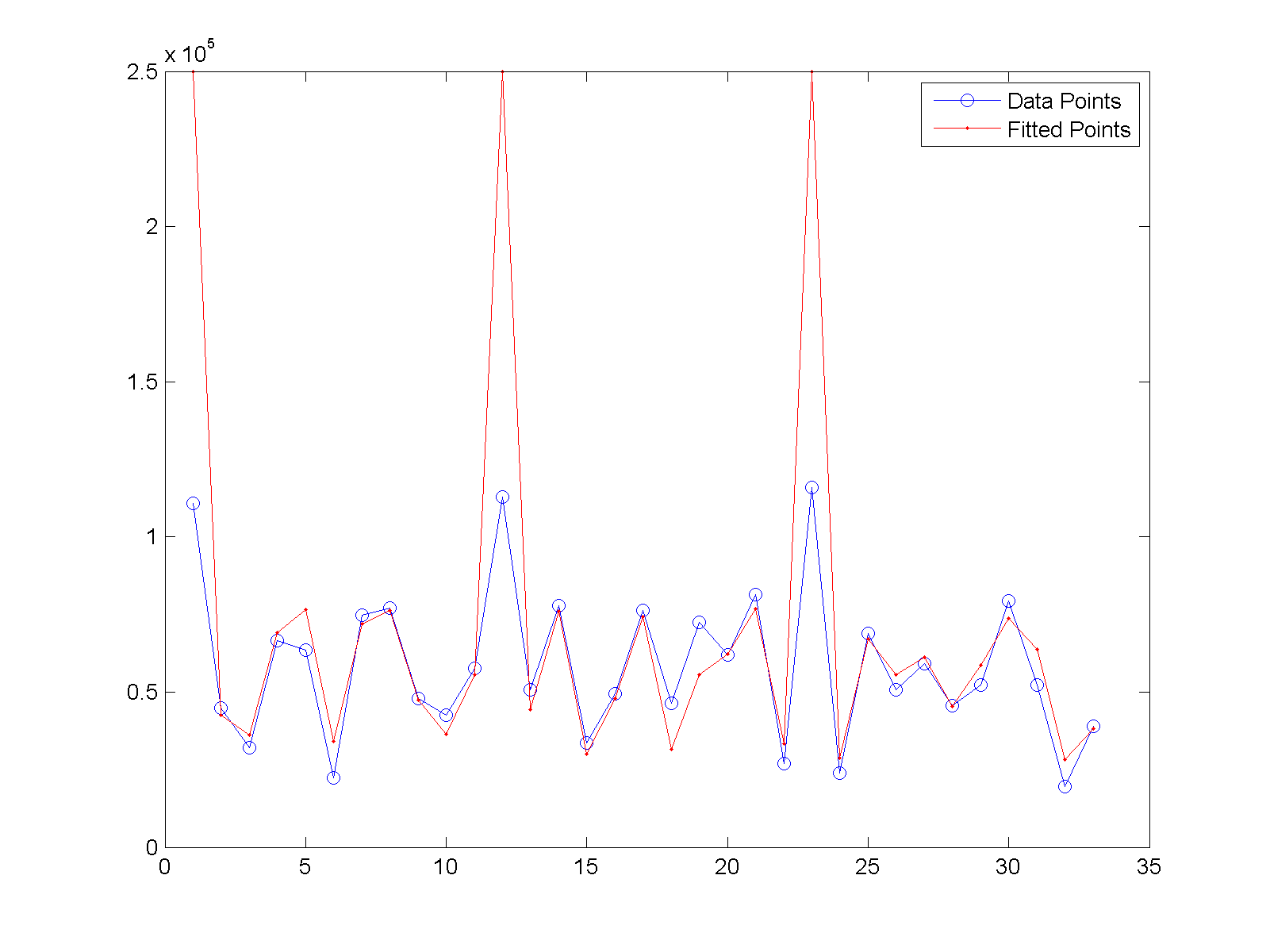
The usual practice is to determine the error associated with the parameters so that the experimental uncertainty can be presented, i.e., standard deviation in the data. In other words what is the magnitude of typical residual? There are several ways to calculate it. However, the statistically optimal measure is the residual standard deviation or standard error of estimate (SEE). Given that each data point carries equal weight or same actual error associated with it,

*N* is the total number of data points

*P* is the number of parameters

is the residual at each data point

In your case the residual sum is 7.4024e+09 with *P*=5 and *N*=33, thus *SEE* is 1.6260e+004. I am not sure whether anything exists to describe a relationship between the data standard deviation and RESNORM. Below is the fitted points for your case



However, with the given range of standard deviation the expected sum of squared difference would be between 0.825e+009 to 1.1880e+009.

For my case I am getting f value as 0.2215 which falls between [0,1]. In your case you can constrain it to be between 0 and 1.

**Q1.1.2**

The new function with constrains

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

S0 and d values are squared and then square root is applied to make sure S0 and d are always positive. Absolute value of f is used to constrain the values of f between 0 and 1. No constraints are imposed on theta and phi, since the trigonometric conversion automatically takes care of these values. For a voxel

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

Using fminunc and BallStickSSD functions the parameter\_hat values are

2.5000e+005

0.0022

0.2215

-0.1050

-1.0437

Using fminunc and BallStickSSDC functions the parameter\_hat values are

2.5000e+005

0.0022

0.2215

-0.1050

-1.0437

which are same since all the conditions are met. The RESNORMs for both of them are

5.7508e+010

However, with another voxel

Avox = dwis(:,42,62,25);

Using fminunc and BallStickSSD functions the parameter\_hat values are

2.5000e+005

0.0013

-0.1155

-0.8801

-0.7736

and the RESNORM is 6.4298e+010

Here, the f value is negative. If constraints are applied on these with BallStickSSDC function, the new parameter\_hat values become

2.5000e+005

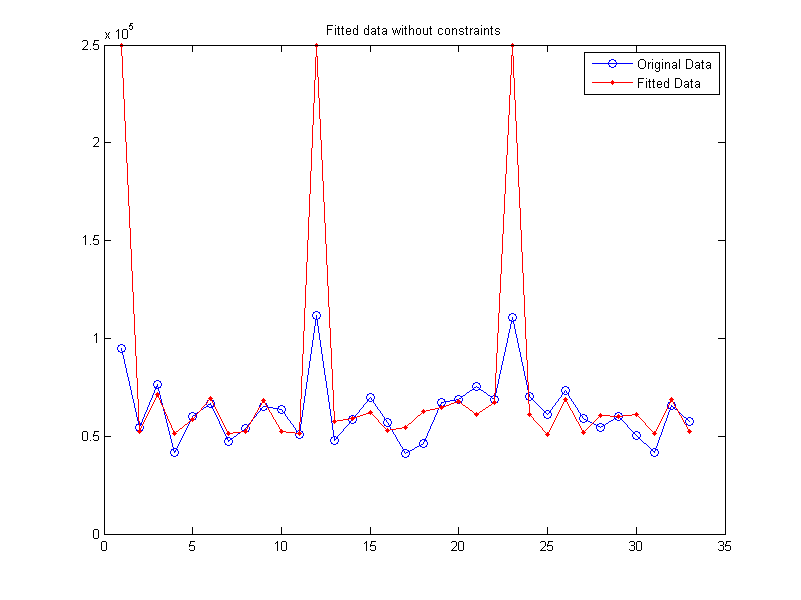
0.0014

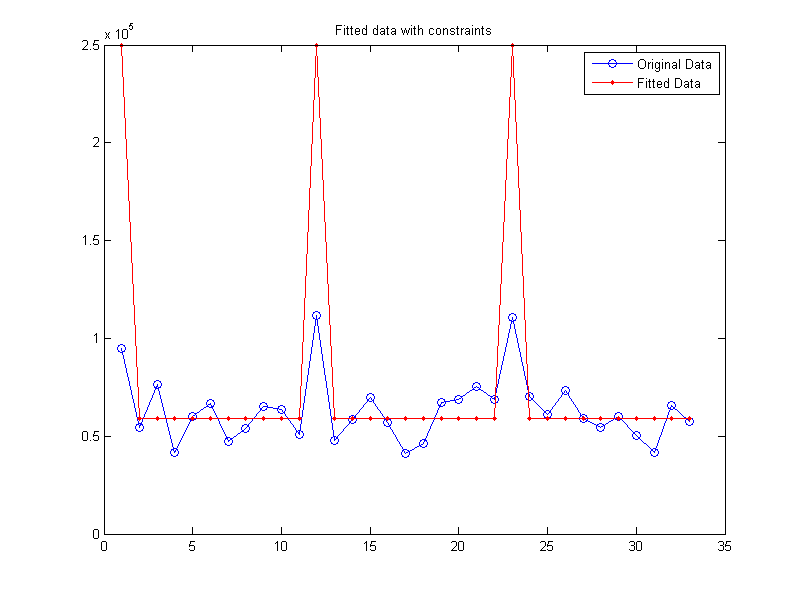
1.4980e-015 ; f value is between 0 and 1.

8.7603e-004

-0.2075

and the RESNORM is 6.5613e+010. The RESNORM value without constraints is lower since it tries to fit with the noisy data. Both the fits are shown below.





**Q1.1.3**

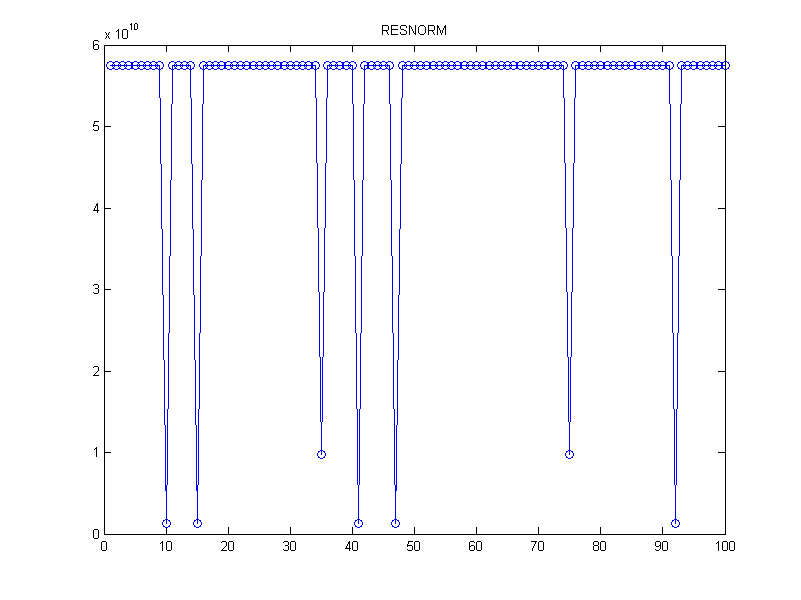
I included perturbation using normal distribution only for S0. However, since other parameters are very small in magnitude, perturbation using fminunc is not allowed as it returns

??? Error using ==> roots at 28

Input to ROOTS must not contain NaN or Inf.

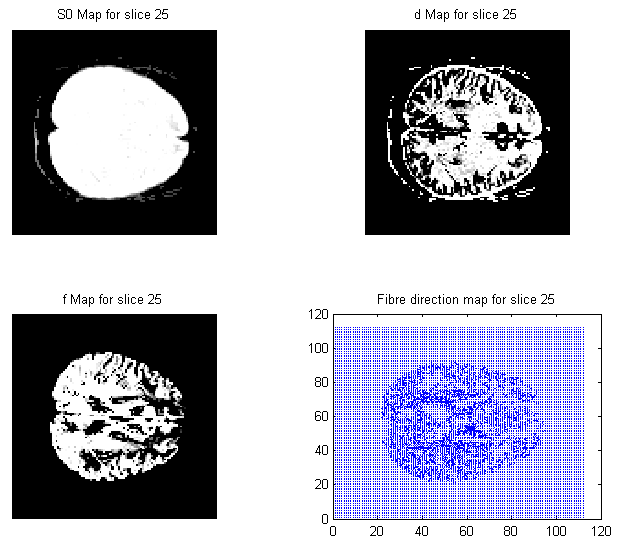
This error means that an operation in the code generated a NaN or Inf rather than a finite number. There are many reasons why this may be, including an unbounded objective function to fminunc or simple errors such as divisions by 0. If you want to add bounds to your coefficients, use fmincon instead of fminunc. If you want to know exactly where the NaN or Inf was generated, use dbstop if naninf in the beginning of your code.

With 95% confidence level and 10 confidence interval, the required sample size would be approximately 100. The RESNORM for 100 samples are shown. Since, just one sample gives the smallest value of RESNORM, it cannot be global minimum.



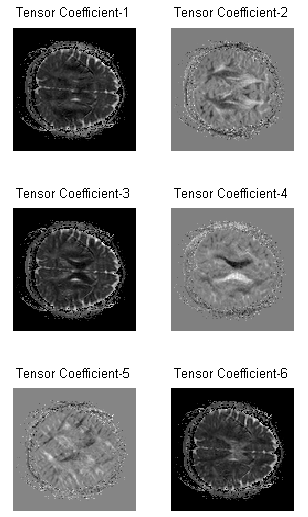
**Q1.1.4**

Since fminunc function displays errors as described in the previous section, fmincon function was used instead to generate maps of S0, d, f and fibre directions as shown below.



**Q1.1.5**

I was not very clear about this question as I don’t have the lecture notes mentioned in the question available. However, matlab code is provided to estimate diffusion tensor. One of the examples is given below



**Q1.1.6**

