STAT 840 A3

Question a

In this question, we will do inference on Dyestuff dataset in R package lme4 using a variance component model. The dataset was motivated to understand how H-acid contributes to the between-batch variation of the dyestuff made from it. The dataset records an attribute of interest of 6 batches of dyestuff made from different sample H-acid, with the batch size being 5. The attribute of interest is called "yield", whose meaning is beyond our scope here. Hence, the model setting here is $Y \sim N(\mu, \sigma^2(I_{30} + \tau ZZ^T))$, where $\mu \in R^{30}$ is a mean vector where the entries in the same batch share the same value, and

$Z \in R^{30\times 6}, Z_{ij} = 1$ if and only if the i'th object belongs to the j'th batch, otherwise it's 0.

(a)

First, we need to get the parameter space of the model, so that we know over which we are able to search for our maximum likelihood estimate (MLE). We showed before that for such a variance component model, the parameter space is $(-1/\lambda_m, +\infty)$, where λ_m is the smallest eigenvalue of matrix $Z^T Z$. This can be done easily using SVD of Z and find the smallest square singular value, and that would be λ_m. The computational result shows that $\lambda_m = 5$, so the parameter space should be $(-1/5, +\infty)$.

The answer can also be easily derived analytically, since the matrix Z's columns' nonzero entries' positions are mutually exclusive, so the cross product matrix only has its diagonal entry being nonzero and always 5, which is the inner product of the column itself. Here is a brief proof:

First, define the batches as $G_j, j = 1, 2, ..., 6$ as mutually exclusive sets, each containing 5 integers between 1 to 30 indicating the object number in the batch.

Then, for any column $j = 1, 2, ..., 6$ in matrix Z,

$$c_j = \sum_{i=1}^{30} 1_{[i\in G_j]} e_i$$

Therefore,

$$Z^T Z = \begin{bmatrix} c_1^T \\ \vdots \\ c_6^T \end{bmatrix} [c_1, ..., c_6]$$

$$(Z^T Z)_{ij} = c_i^T c_j = \sum_{k=1}^{30} 1_{[k\in G_i]} e_k^T \sum_{t=1}^{30} 1_{[t\in G_j]} e_t = \sum_{k\in G_i} e_k^T \sum_{t\in G_j} e_t = \sum_{k=t\in G_i, G_j} 1$$

Therefore, the entry value is 0 if it's not on the diagonal, and it is 5 is it's on the diagonal:

$$(Z^T Z)_{ii} = \sum_{k \in G_i} 1 = |G_i| = 5$$

Thus, we know that $Z^T Z = 5I_{30}$, which means that its eigenvalues are all 5, which gives the lower bound of the variance component model $-1/5$.

In conclusion, the parameter space for the model setting is that $\tau \in (-1/5, +\infty)$.

(b)

First, we reuse the function in A2 to calculate the NLL for this dataset also. The plot of the NLL is like this:

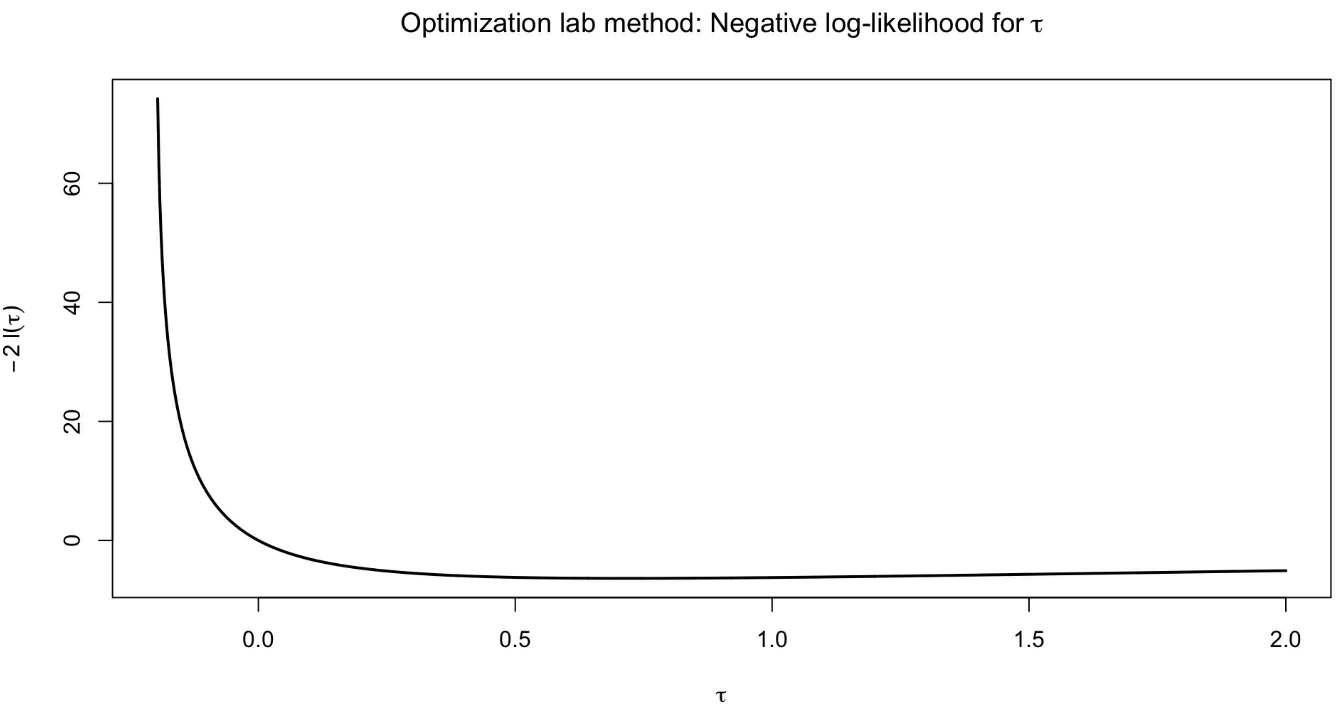

Fig.1 NLL calculated using the algorithm in the optimization lab

We can see that the plot has an obvious left barrier, which is exactly $\tau = -1/5$ as we derived from (a). To see how it behaves when τ is big at very left:

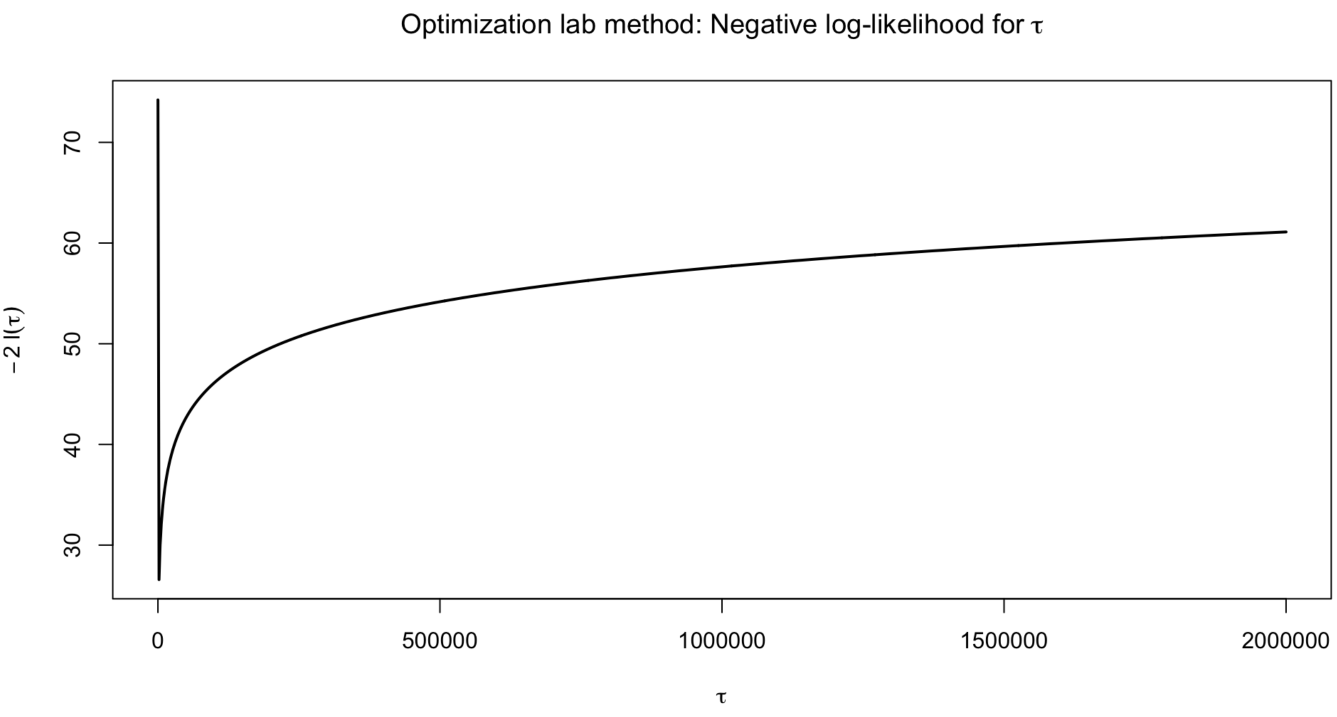

Fig.2 NLL plot with a broader view

The more global pattern plot shows that the NLL drops down from infinity to the MLE very quickly around 0, then goes up very slowly. Nonetheless, we know the actual convex neighborhood around the MLE is not as sharp as it looks like in Fig.2, but very flat according to the local pattern plot in Fig.1. The fact that is of our interest here is that the right part becomes flatter and flatter, which causes the speed-up divergence we will observe in some latter experiment when the wrong direction leads the iterations to the right part. We may expect the NLL to be locally convex around NLL. The local convexity around MLE makes Newton's method work regularly if the initial value is within the convex neighborhood, i.e., close enough to the MLE. The plot also tells us that it's very likely that the NLL has only one local minimizer, and thus the global minimizer. Hence, if we use any numerical methods to find a point where the gradient is 0, it is very likely to be the MLE. Nonetheless, the plot is not globally convex obviously, at least for the right part, which might cause some problems with Newton's method when the initial value is not close enough to the MLE. The global plot seems quasi-convex in shape, since for any 2 points on the function, the function curve between them seems to be not above the larger end point. This property makes Newton's method usable, but with some potential risks that we should resolve in question (c).

(c)

In this question, we employ Newton's method to solve the MLE. As we discussed in (b), Newton's method can be executed in the case where the function looks quasi-convex, and the vanilla one is expected to succeed when the initial value is close enough to the MLE. According to Fig.1, $\tau \in (0.5, 1)$ seems to be a locally convex neighborhood around the MLE, so we can first try to set the initial values within that interval and run vanilla Newton's method:

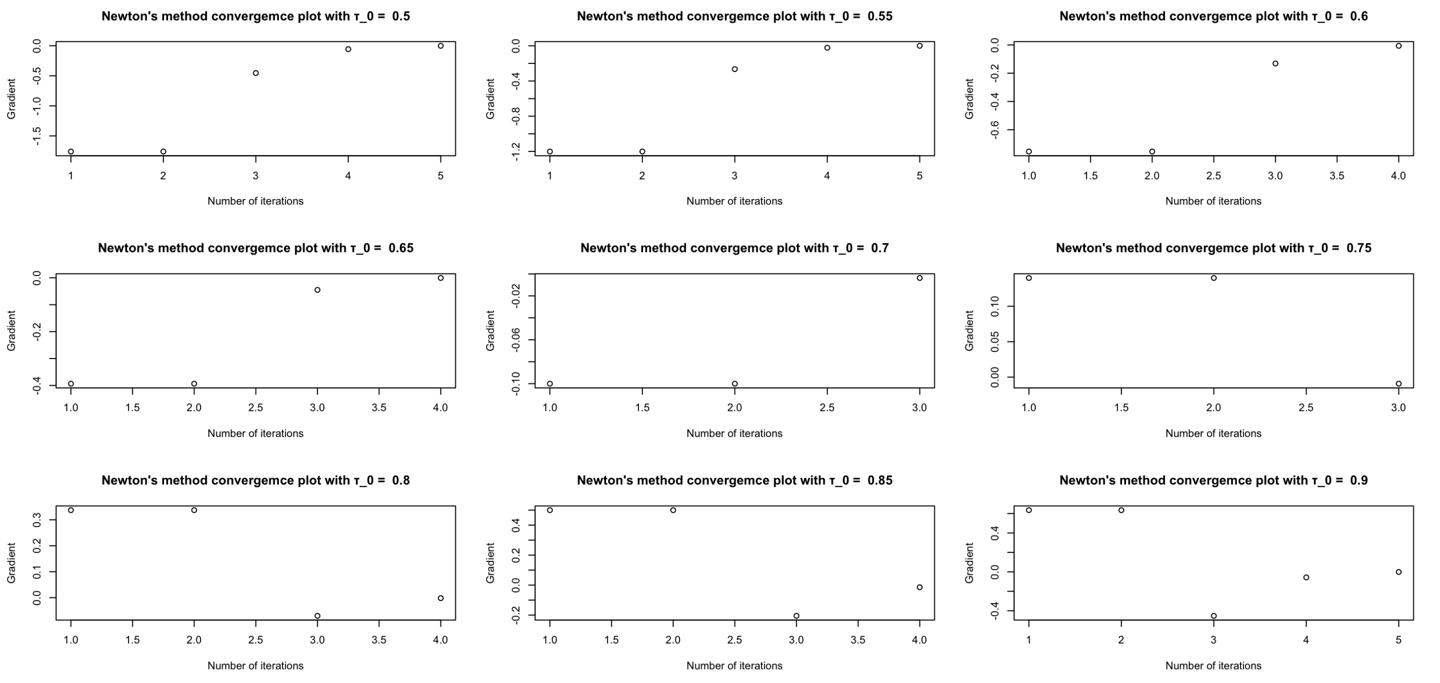

Fig.3 vanilla Newton's method number of iterations vs. gradient value with initial values in the locally convex neighborhood of the MLE, i.e., [0.5, 1]

The convergence plot in Fig.3 shows that when we input a series of points: 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9 as the initial value respectively, the vanilla Newton's method indeed always converges within 5 iterations. The MLEs derived using those initial values are all around 0.7196531, which is robust and reliable according to the NLL plot in Fig.1.

Nonetheless, when we try larger τ values to reach the non-convex right part, it fails to converge to the MLE:

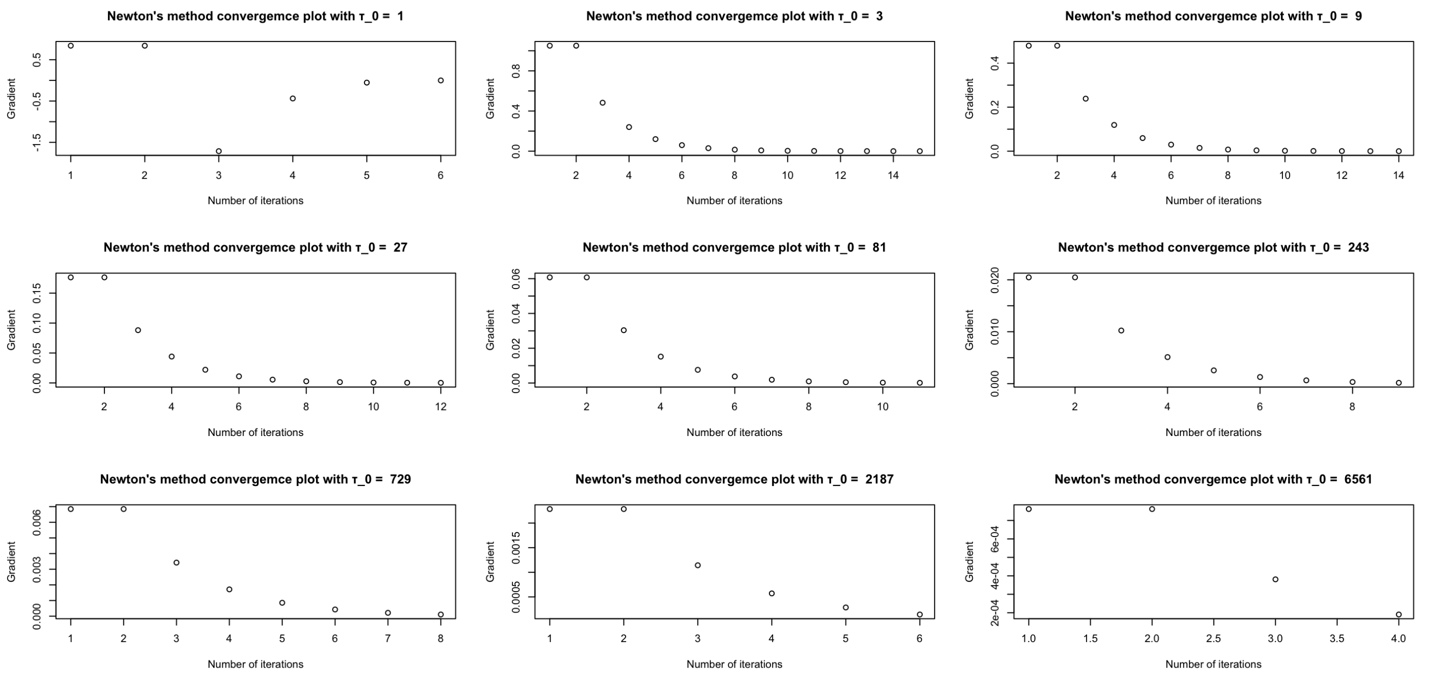

Fig.4 vanilla Newton's method number of iterations vs. gradient value with initial values outside the locally convex neighborhood of the MLE

Fig.4 was generated with initial values 3, 3^2, ..., 3^8. They seem to converge finally, but not to the correct MLE except the first one starting from 3. Their results are something like 84302.91, 93478.77, 58088.76, etc., and none of them are the MLE around 0.7196531. In that case, I suggest that we call them divergent.

Then, we try the initial values around the left barrier from -0.18 to 0.62. The good news is that, even though I didn't implement any reflection operations, I haven't met a case where the MLE breaks the left barrier of the parameter space:

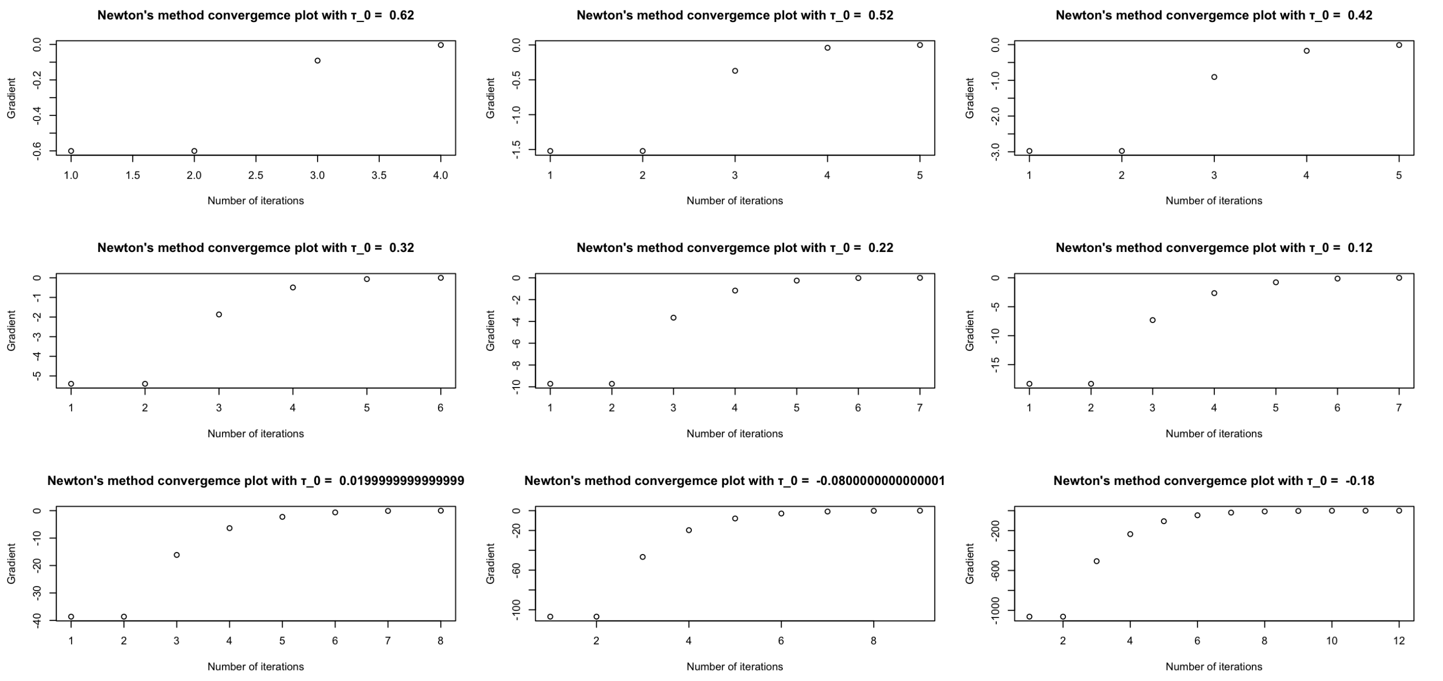

Fig.5 vanilla Newton's method number of iterations vs. gradient value with initial values near the left barrier of the parameter space

We will see that the problem should actually be a result of the direction correcting measure we will take later. The only problem here might be that they converge slower: it usually takes them more than 5 iterations to converge, compared to the less than 4 iterations when the initial value is close to the MLE. Nonetheless, they all converged to the MLE, i.e., 0.7196465 at the end.

Therefore, we may focus on solving the divergence problem. We can clearly see that the right part of the NLL is non-convex, which causes the generation of some negative Hessian values. The negative Hessian value leads the update towards the ascending direction that is opposite to the MLE. Then, we take a remedial measure: Take the absolute value of the Hessian spectrum so that it never leads to an ascending direction.

After I modified the vanilla Newton's method to force the Hessians to be positive, the first and third sets of experiments are replicated with no problems and they all successfully converge to the correct MLE:

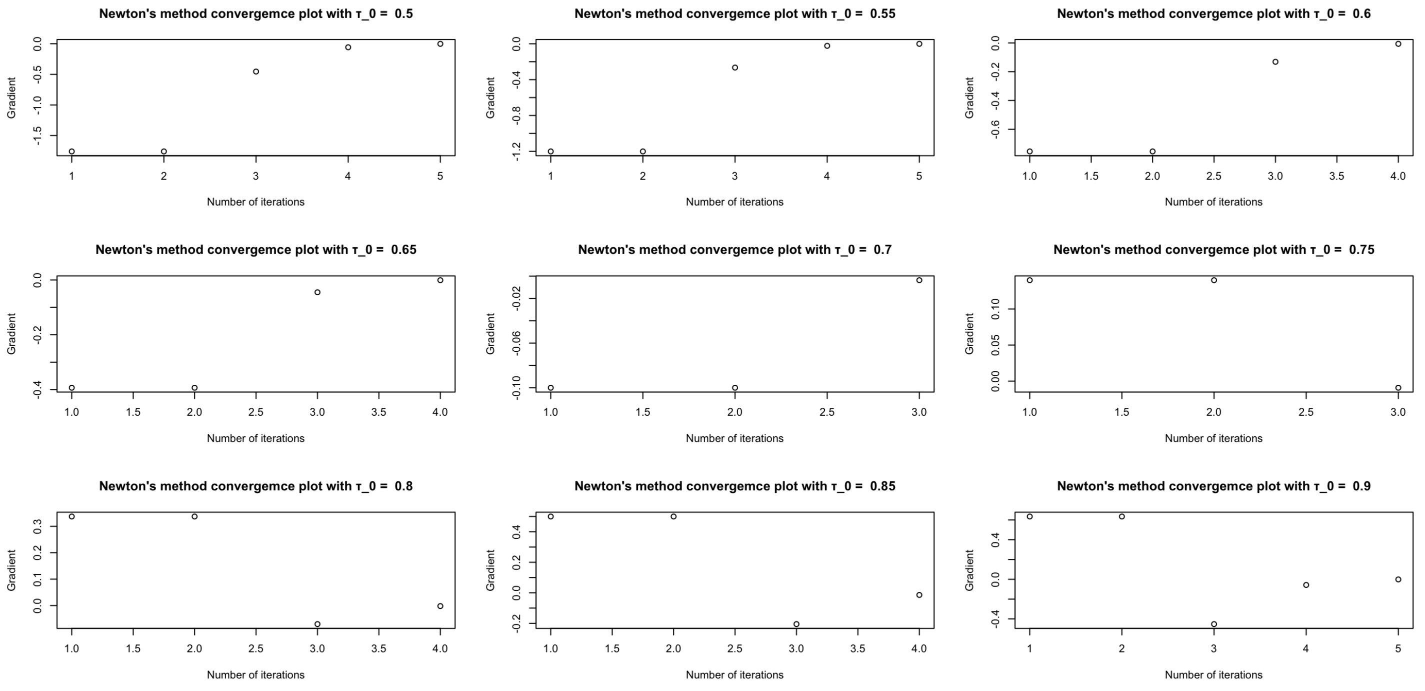

Fig.6 Re-run of experiment 1 after taking Hessians' absolute value

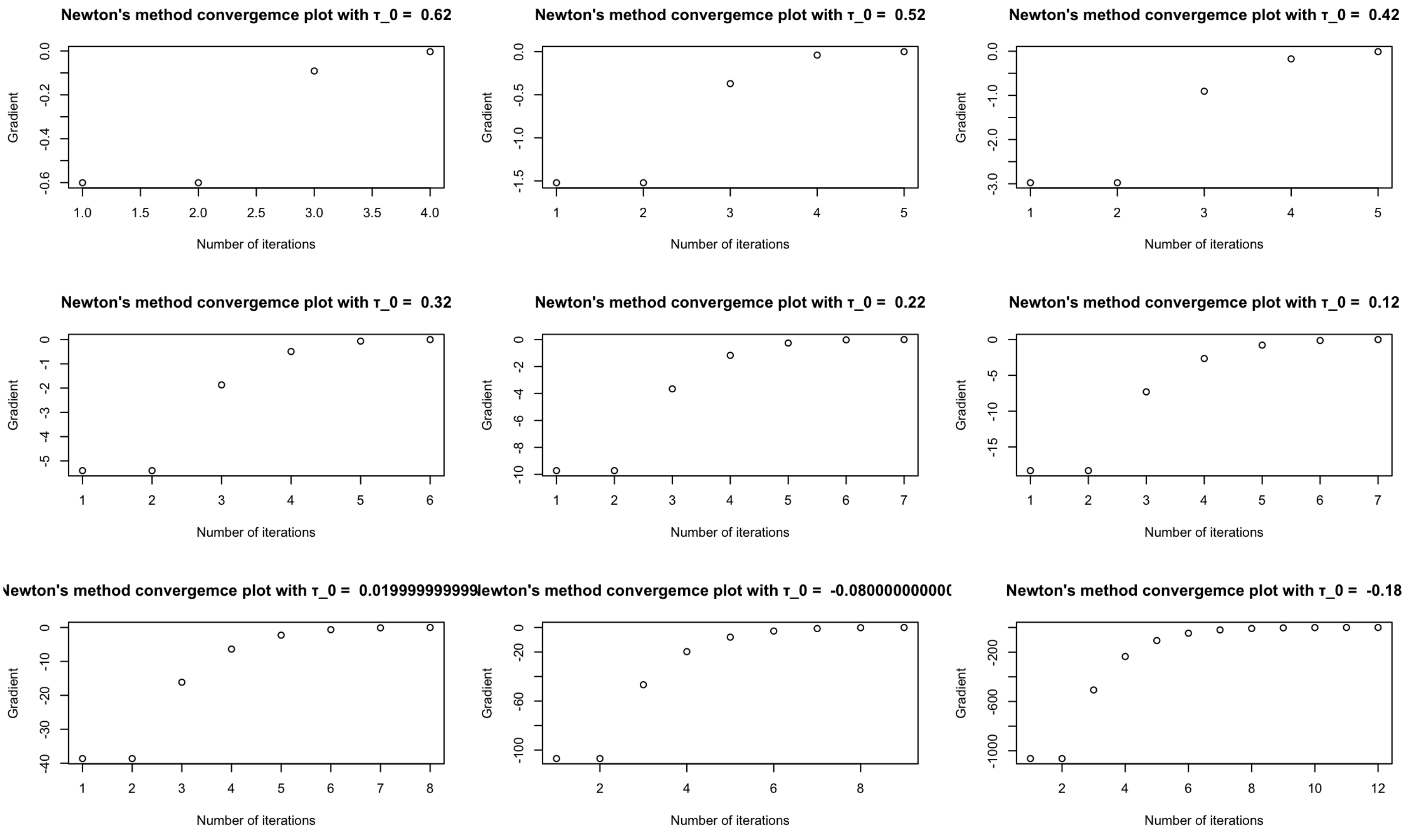

Fig.7 Re-run of experiment 3 after taking Hessians' absolute value

However, the second set of experiments (the ones with initial values in the very right region of the NLL) still failed, and even worse: NaN values were generated. I tracked the debugger to verify that it was because, even though the direction was indeed towards the MLE, the step size was too large due to the too flat gradient, the iterations broke the left barrier of the parameter space. Starting with 3, the third iteration went to -2.641528, which was already outside of the parameter space. The following iterations just went even farther, which caused the failure of the iterations. The reason is that the right non-convex part's curvature is very small, and the NLL (and thus the gradient) becomes flatter and flatter, which makes the Hessian values closer and closer to 0. Therefore, the inverse of the Hessian values gives larger and larger step sizes, leading the iterations farther and farther against the MLE. Therefore, we must take the second remedial measure: Halve the step size if the iteration reaches the left barrier or generates a NaN value.

After implementing the second measure, we finally make the algorithm succeed on the large initial values (i.e., experiment 2):

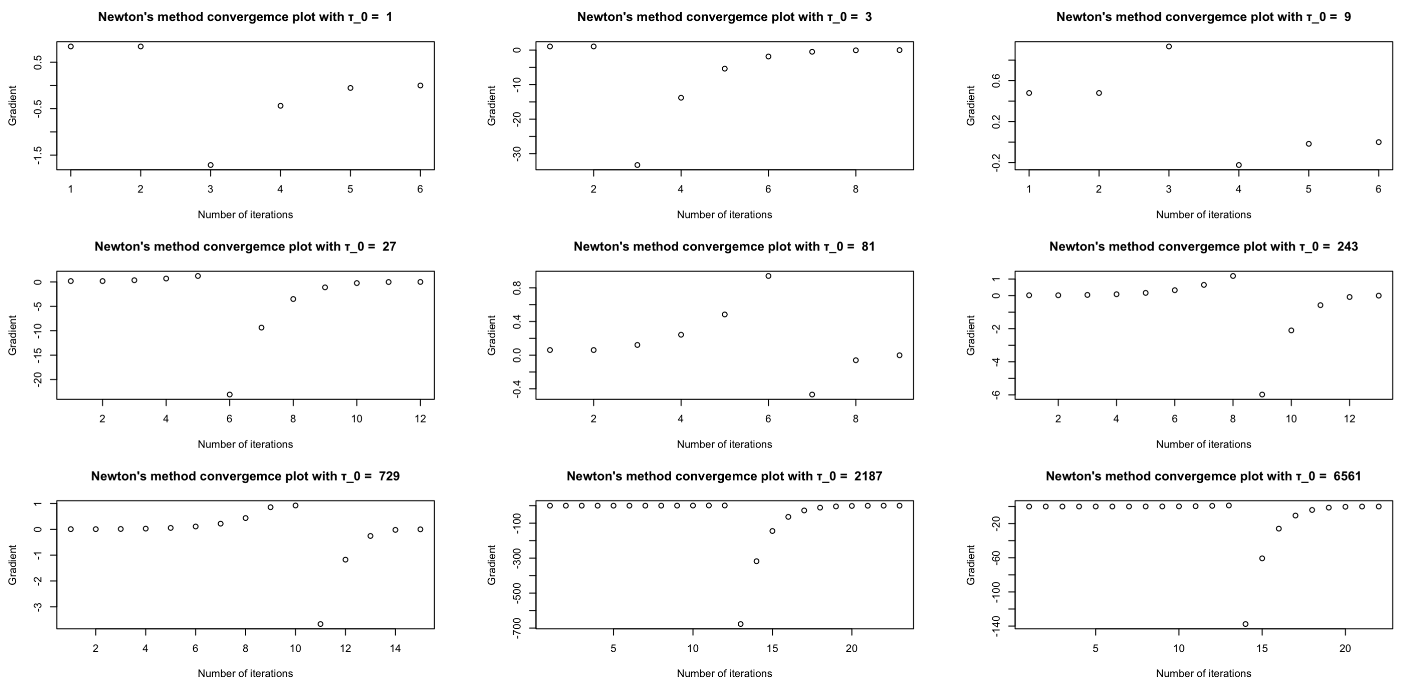

Fig.8 Re-run of experiment 2 after taking Hessians' absolute value and halving the too-large invalid step sizes

According to Fig.8, the algorithm, even though takes more iterations to converge, i.e., at most more than 20 iterations, they all converge to the correct MLE. We can see strong evidence in the plots that they all once changed their convergence directions at some point, which then proved to be the correct one according to the result.

Then, just in case that the algorithm fails in the other 2 cases, we rerun experiment 1 and 3 for the near-MLE region and near-barrier region:

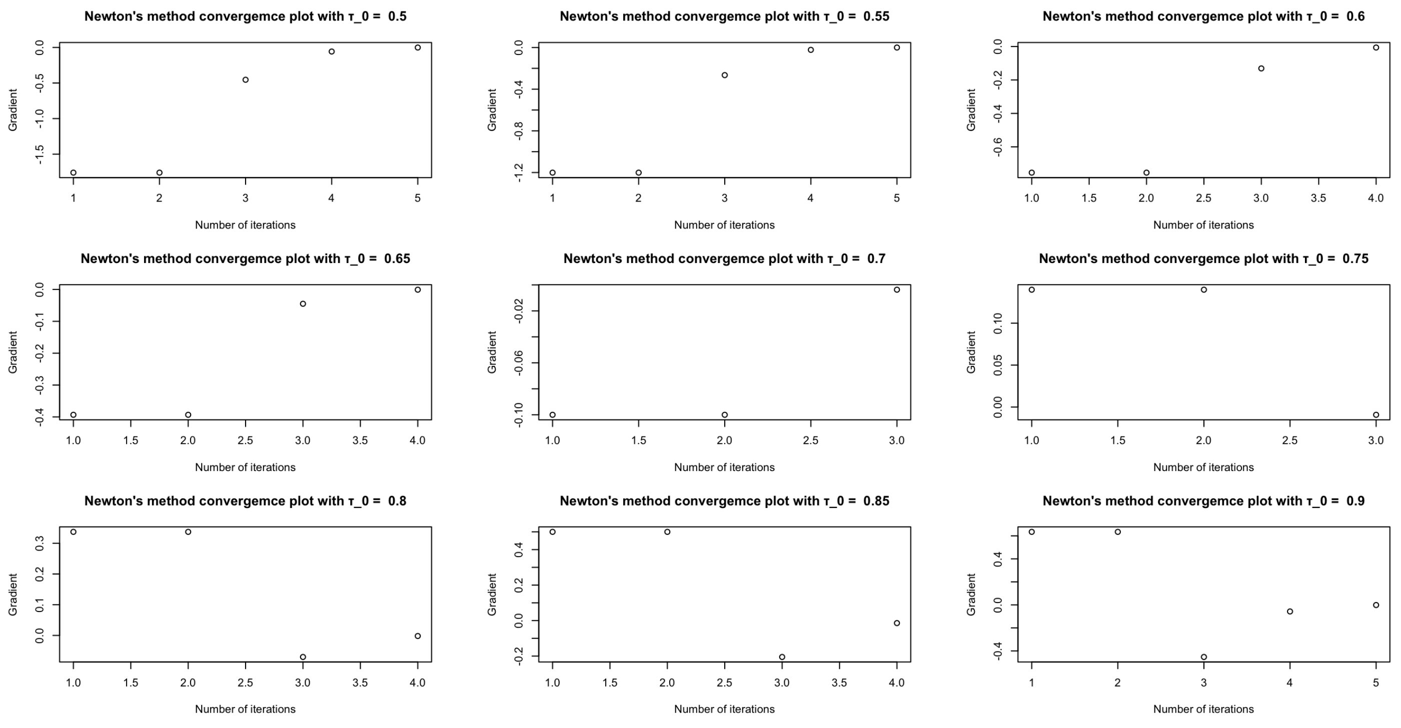

Fig.9 Re-run of experiment 1 after taking Hessians' absolute value and halving the too-large invalid step sizes

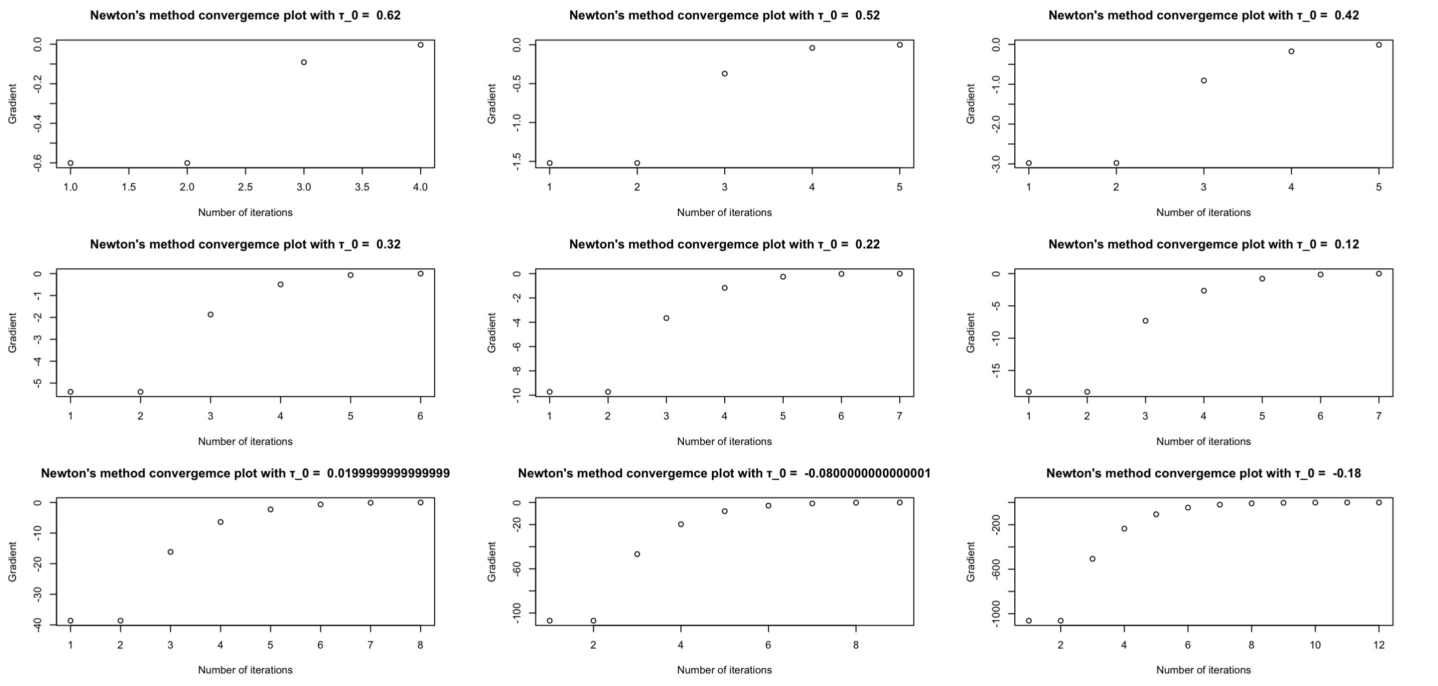

Fig.10 Re-run of experiment 3 after taking Hessians' absolute value and halving the too-large invalid step sizes

Not surprisingly, the algorithm succeeds without any performance loss in the near-MLE and near-barrier regions.

To more harshly test the algorithm, I will generate a very variable set of initial values of size 1000 to test their convergence. The result will show whether they converge to the MLE and how many iterations they take respectively. First, I randomly generate a set of uniformly distributed initial values on $[-0.18, 3000]$ with a size of 10000. The empirical density of the initial values is:

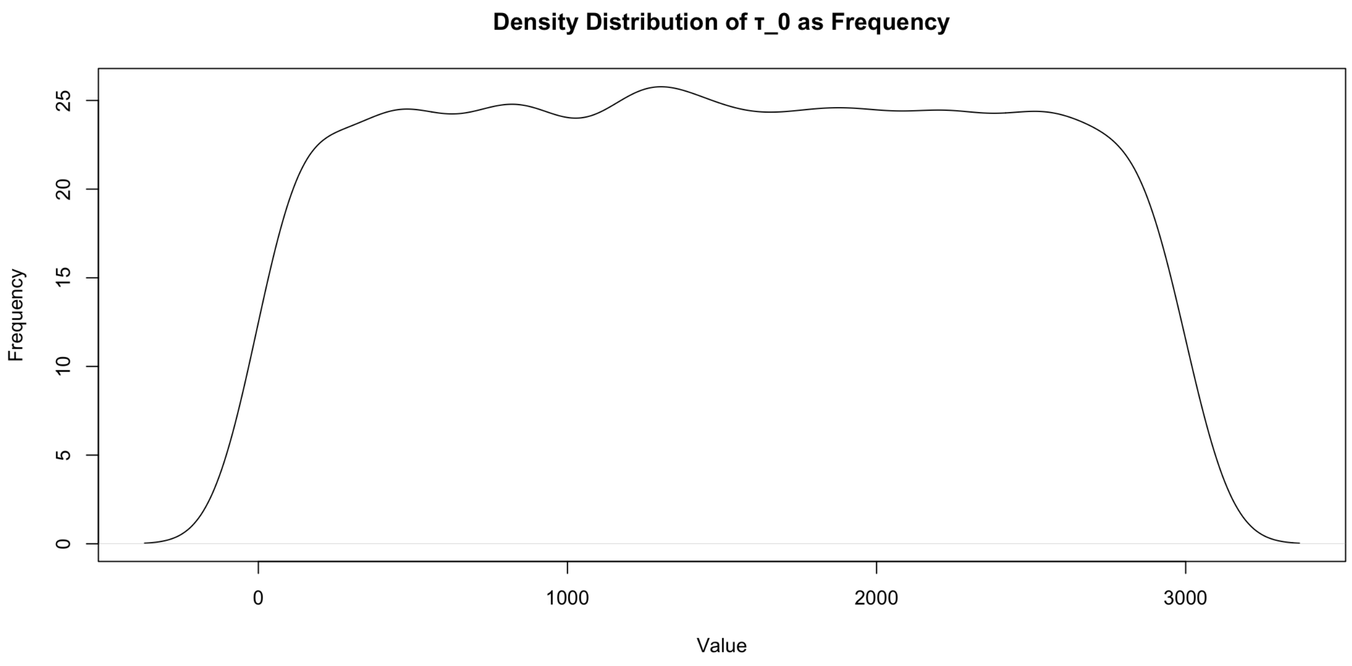

Fig.11 Empirical density function of the generated initial values

The initial values are sampled from -0.18 to 3000 with enough range and sampling density to make sure that as much as initial values are tested for the robustness of the modified Newton's method. Then, the range of the optimized MLE based on the 10000 different and variable initial values is $[0.7196328, 0.7196532]$ without NA values. The plot visualization is here:

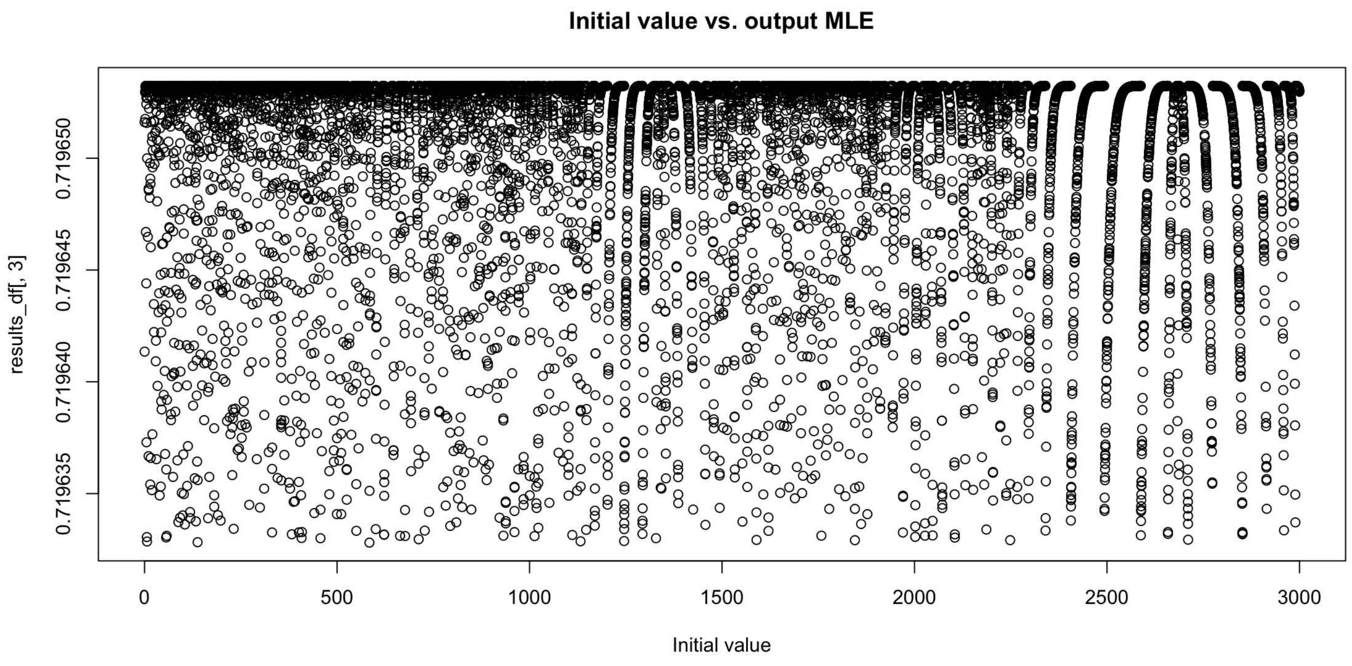

Fig.12 Initial value vs. output MLE based on 10000 different initial values ranging from -0.18 to 3000

It means that they all converge to the MLE with very little error. Actually, the error width is because I set a tolerance parameter, so they would stop once the error is within the tolerance (i.e., the absolute gradient is smaller than the parameter). Then, we can check the number of iterations they take to converge:

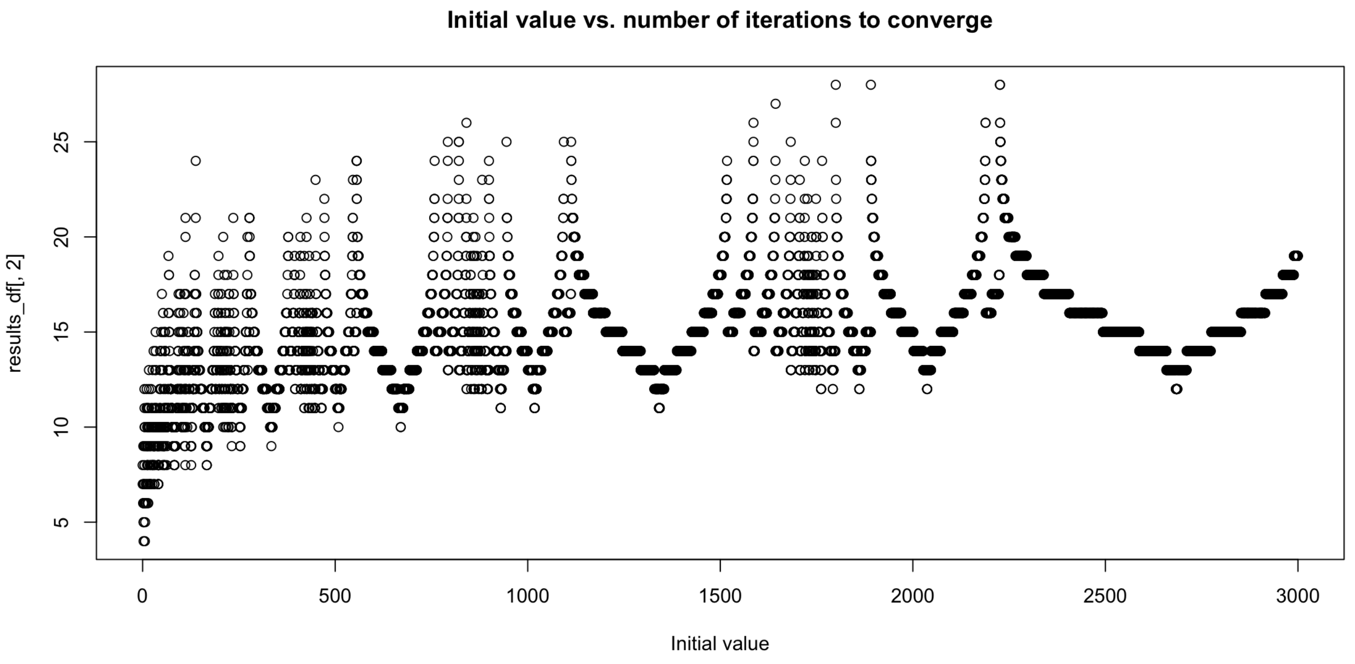

Fig.13 Initial value vs. output MLE based on 10000 different initial values ranging from -0.18 to 3000

According to Fig.13, they all converge within 30 iterations, with a range of $[4, 28]$. The convergence speed doesn't seem to have a pattern as the initial value goes larger, but it's always the fastest near the MLE. The resistance of convergence speed w.r.t. how far the initial value is to the MLE benefits from the Hessian information provided by Newton's method as we discussed before.

In conclusion, the MLE of the model is $\hat{\tau} \approx 0.7196465$ using modified Newton's method. I implement 2 remedial measures based on the vanilla Newton's method. First, I take the absolute values of the Hessian to make sure that it doesn't diverge to the opposite of the descending direction on the right part of non-convex parameter space. Then, I keep halving the step size when it leads the parameter to the outside of the left barrier until the parameter stops somewhere in the valid parameter space. The latter measure is taken to solve a follow-up problem brought by the first measure according to my trial. The algorithm is expected to converge well, because the NLL is quasi-convex and I correct basically all the problems with Newton's method from the violation of convexity. Indeed, I tested 10000 uniformly generated initial values from -0.18 to 3000, and the result shows that none of them failed. They all converge to the MLE within 30 iterations, with the fastest one converging in 4 iterations only. The experiment result proved that the algorithm is robust enough when the initial value is between -0.18 and 5, since it even passed a much harder challenge: succeeding with a large number of initial values spanning from -0.18 to 3000. Therefore, the algorithm is fast, stable, and robust to the initial values, and we have known what measures to take and why they work based on the trials and results.

(Side note: What I meant by "Hessian" are just real numbers here, which might be confusing. I just hope to generalize it to the multi-dimensional cases.)

Question b

In this question, we rerun all the previous experiments on dataset Dyestuff2, whose residual variance σ^2 is adjusted to a larger value and almost nothing else changes. Then, if necessary, we modify the procedure of Newton's method to make it robust to initial values, stable, and fast enough.

First, we plot the NLL to see its shape pattern

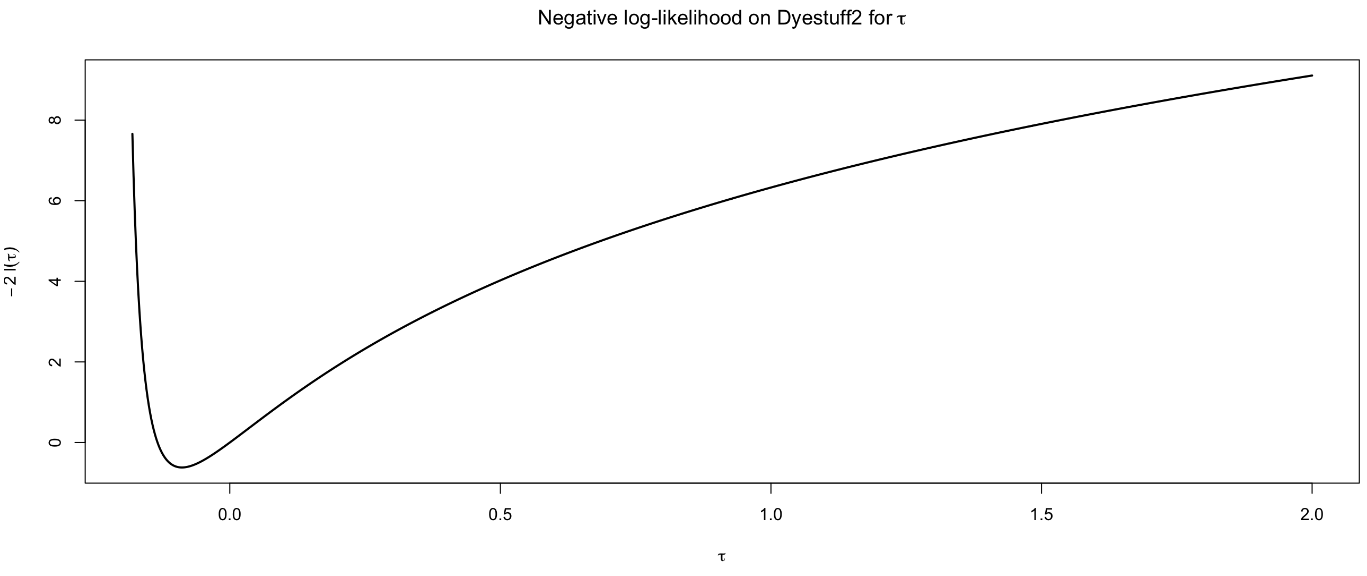

Fig.14 Dyestuff NLL

Compared to the Dyestuff dataset NLL, this one has a negative MLE and the flat area around the MLE becomes much more steep. The shape is still not convex, with the right part being non-convex. The locally convex neighborhood around the MLE still exists, but shrinks to very small compared to the Dyestuff one. The overall shape is still quasi-convex, which makes Newton's method possible to run (at least with some modifications) but prone to some systematic mistakes. There is also a left barrier in the parameter space. The barrier is still $\tau = -1/5$ since the structure of the dataset never changes (and thus so is the Z matrix). Overall speaking, I believe that the safest algorithm devised in part b that does both direction correcting and step size halving can also work well here. We will see it by conducting similar experiments as we did in question a.

First, we try a set of initial values around the locally convex area around the MLE, say $[-0.19, -0.03]$, using vanilla Newton's method:

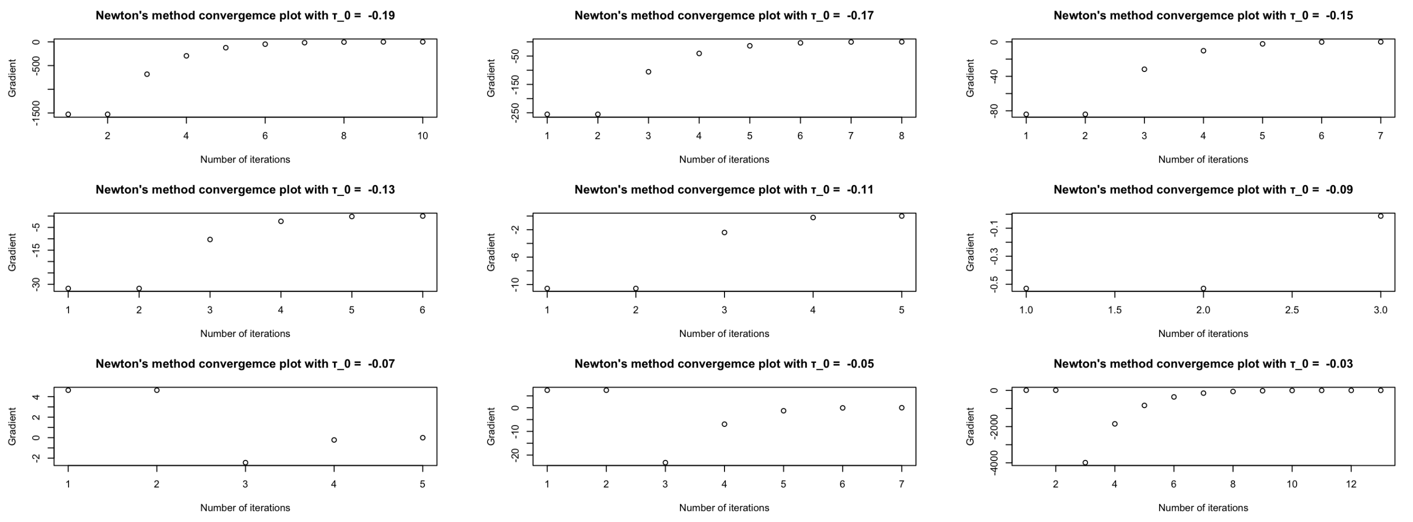

Fig.15 vanilla Newton's method on Dyestuff2: number of iterations vs. gradient value with initial values in the locally convex neighborhood of the MLE, i.e., [-0.19, -0.03]

According to Fig.15, they all converge to -0.08844658 within 15 iterations, which almost tripples the number of iterations needed to converge compared to Dyestuff. It might indicate that when the residual variance is huge, the convergence of the MLE will become slower. I think the reason is that the convex neighborhood of the MLE here is not as flat as the Dyestuff one, whose Hessian inverse is then not as large as the Dyestuff, so the small step size causes the convergence to be much slower. We can also notice that due to the shrinkage of the locally convex MLE neighborhood, the safe initial value interval is much narrower, which makes the model less robust to the initial value since the choice becomes much more picky.

Then, we try a set of initial values on the non-convex area around the MLE, say $\tau \in [0.2, 5.0]$ using vanilla Newton's method:

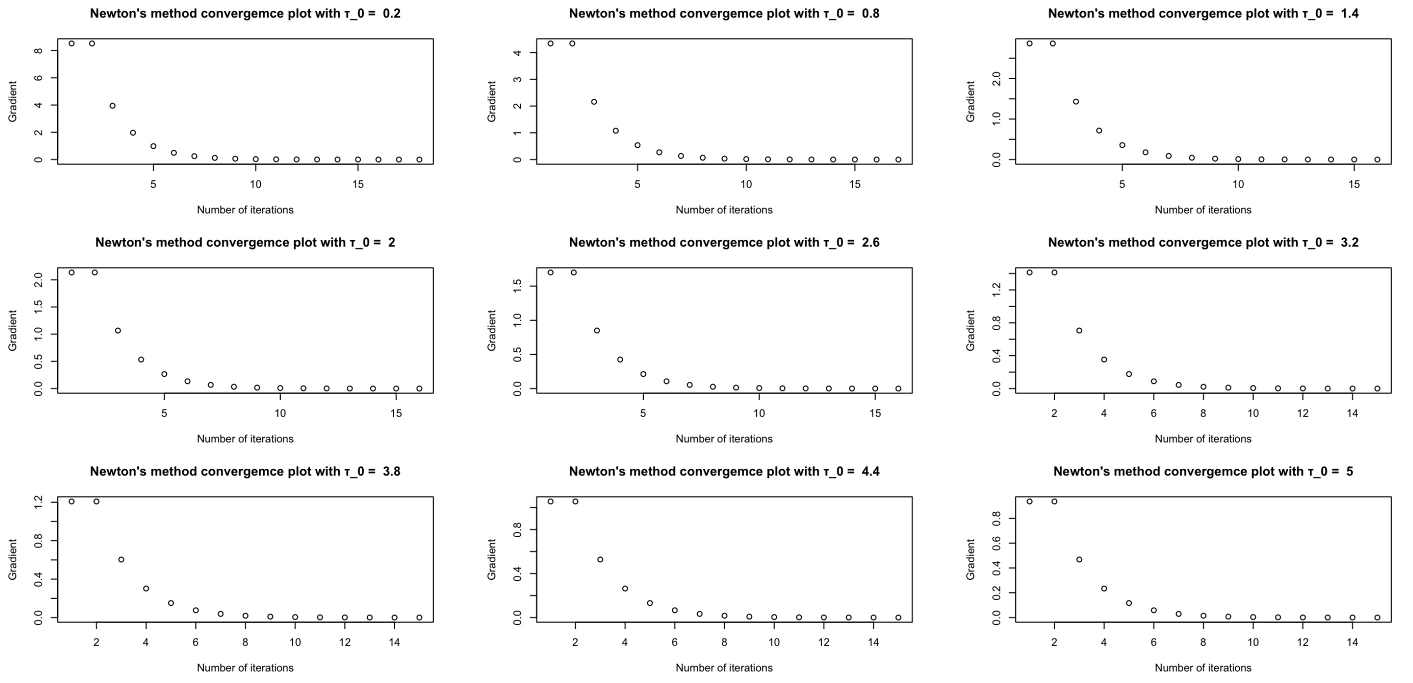

Fig.16 vanilla Newton's method on Dyestuff2: number of iterations vs. gradient value with initial values outside the locally convex neighborhood of the MLE, i.e., [0.2, 5.0]

According to Fig.16, the algorithm fails whenever the initial value is not within the locally convex neighborhood of the MLE. They diverge to some numbers like 63414.92, 89885.3, or 84196.32. The first one even broke the left barrier of the parameter space, giving -65840.07. The result is not surprising due to the non-convex nature of this part of the NLL. The non-convexity yields some negative Hessian spectrum, misleading the iterations towards an ascending direction.

Then, we take the 2 remedial measures again that are exactly the same as we did in part a: 1) take the absolute value of the Hessian to correct their directions, 2) halve the step sizes when the iteration is about to be out of the parameter space, then we rerun the experiment. The re-generated plot is here:

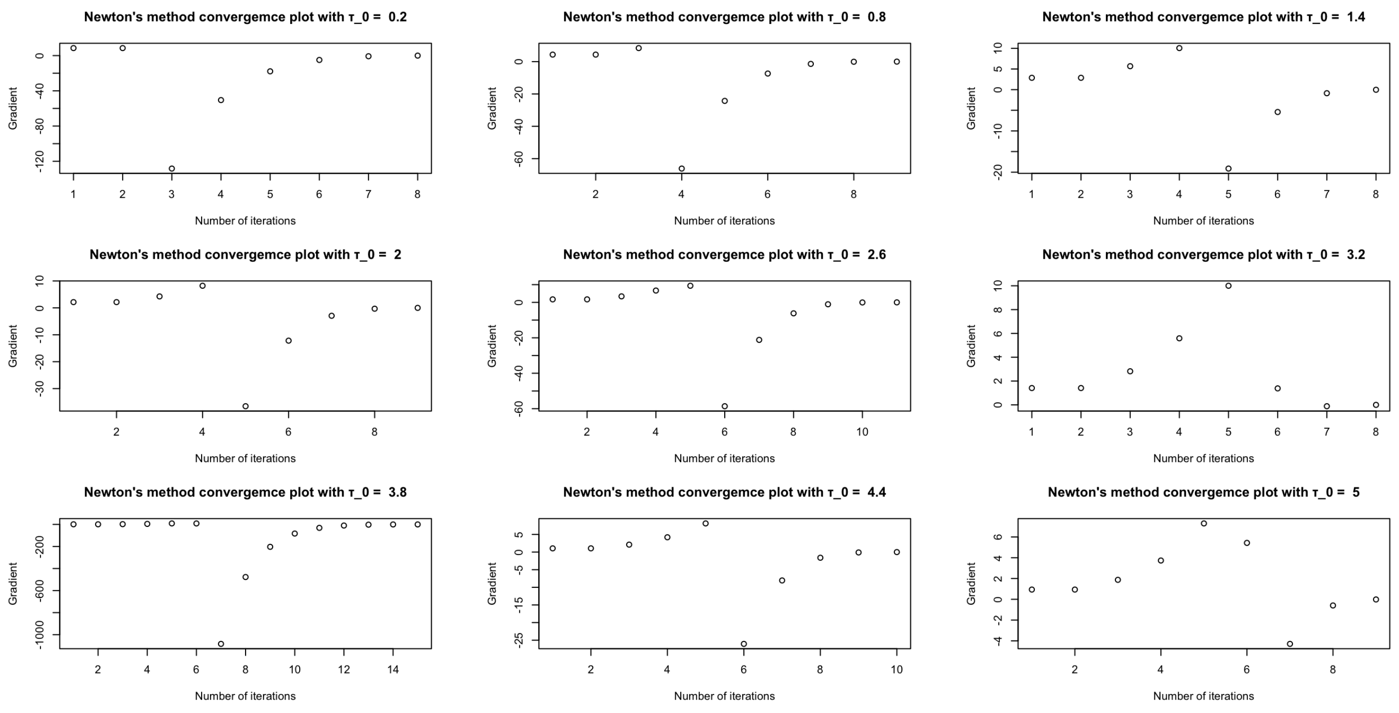

Fig.17 Modified Newton's method on Dyestuff2: number of iterations vs. gradient value with initial values outside the locally convex neighborhood of the MLE, i.e., [0.2, 5.0]

The algorithm converges to MLE within 15 iterations for all initial values this time, even though the values are all in the non-convex interval. To make sure that it still works in the locally convex area, we rerun the first experiment again to verify:

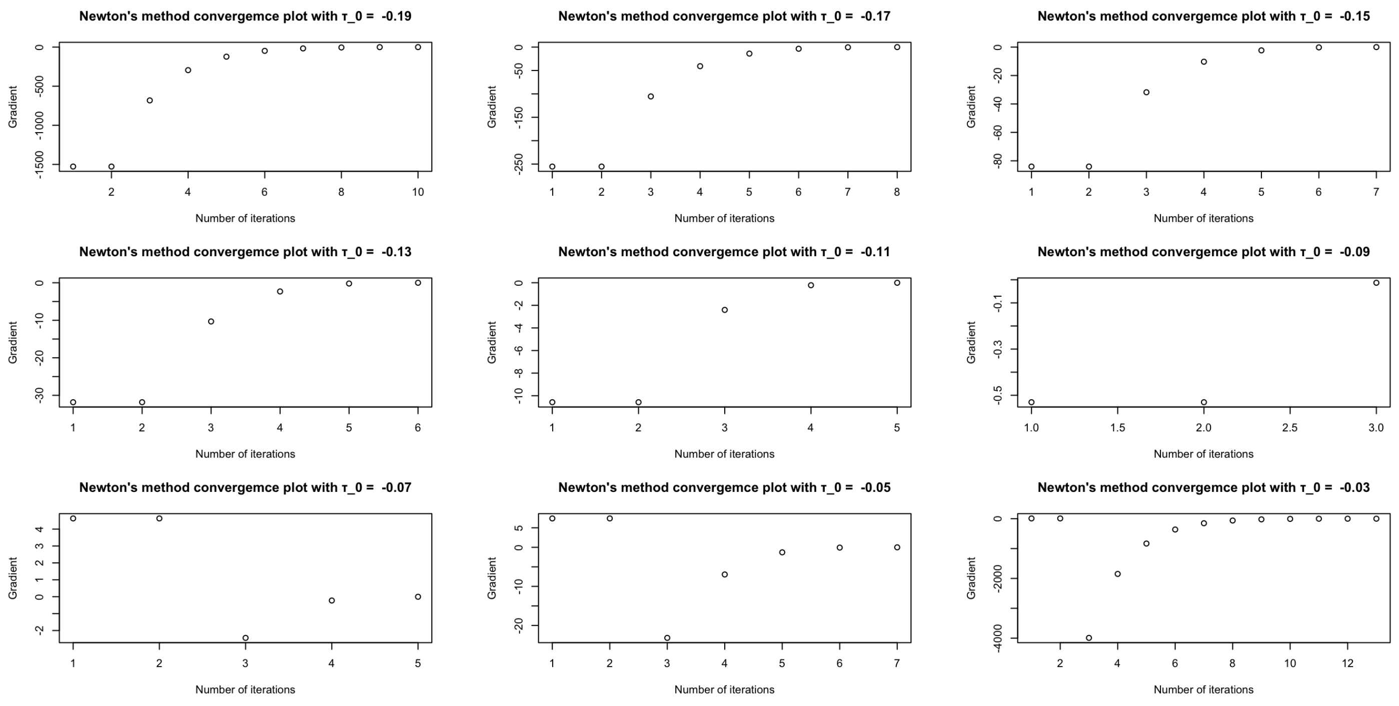

Fig.18 Modified Newton's method on Dyestuff2: number of iterations vs. gradient value with initial values outside the locally convex neighborhood of the MLE, i.e., [-0.19, -0.03]

According to Fig.18, the algorithm still succeeds in the nice interval, without any difference in performance, which is as expected, since it doesn't activate the modified functionality on the interval.

Finally, we test the algorithm over a big set of 10000 randomly sampled initial values from -0.18 to 5 to validate its robustness to the initial value and its performance. The sampled initial value has this empirical density:

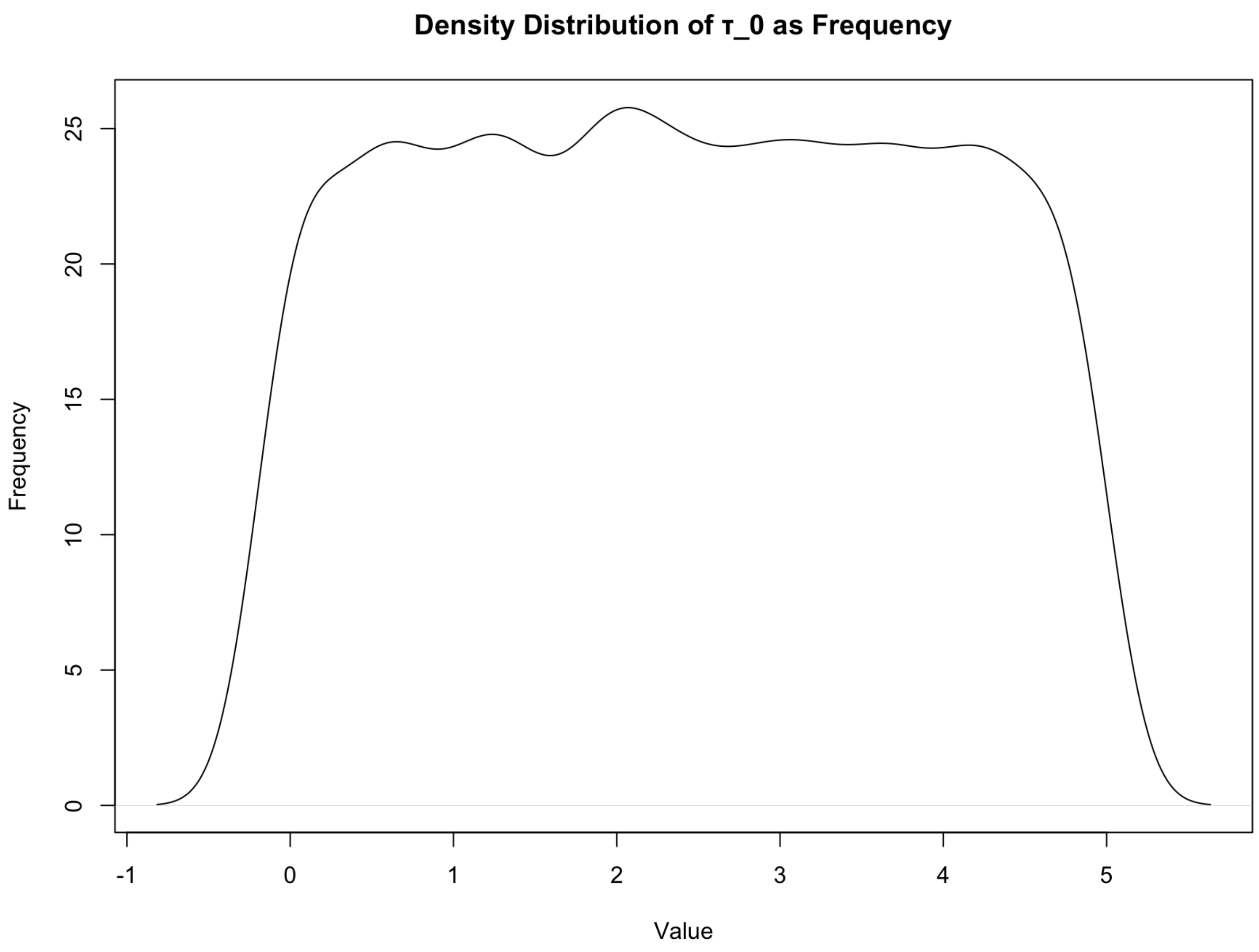

Fig.19 Empirical density function of the generated initial values

The algorithm's output with different initial values is shown in this plot:

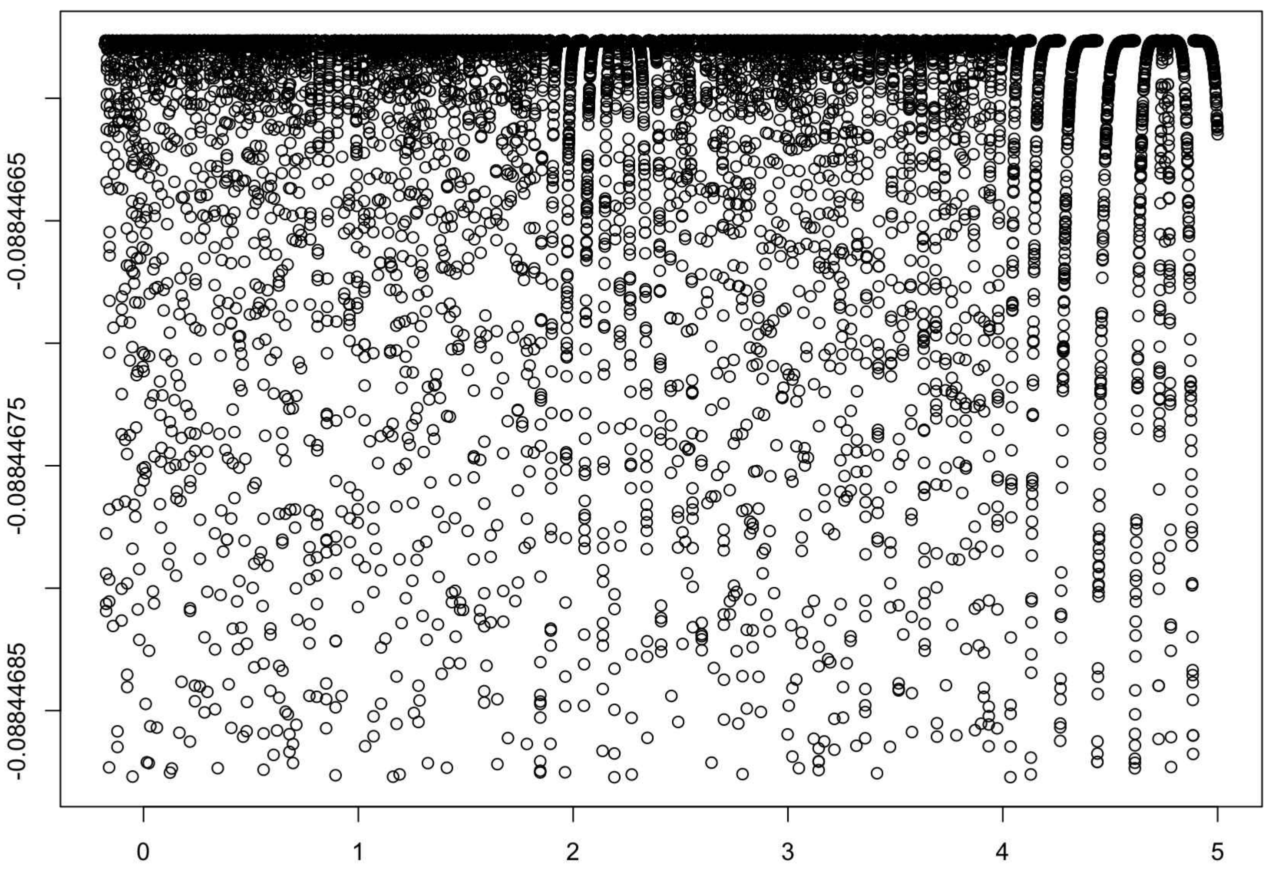

Fig.20 Initial value vs. output MLE based on 10000 different initial values ranging from -0.18 to 5

According to Fig.20 and my R output, the algorithm converges to the MLE for all the tested 10000 initial values between -0.18 to 5, with a slight error width no larger than 3e-7, which is caused by my set error tolerance as one of the stopping criteria of the algorithm. The convergence speed with different initial values is shown in Fig.21:

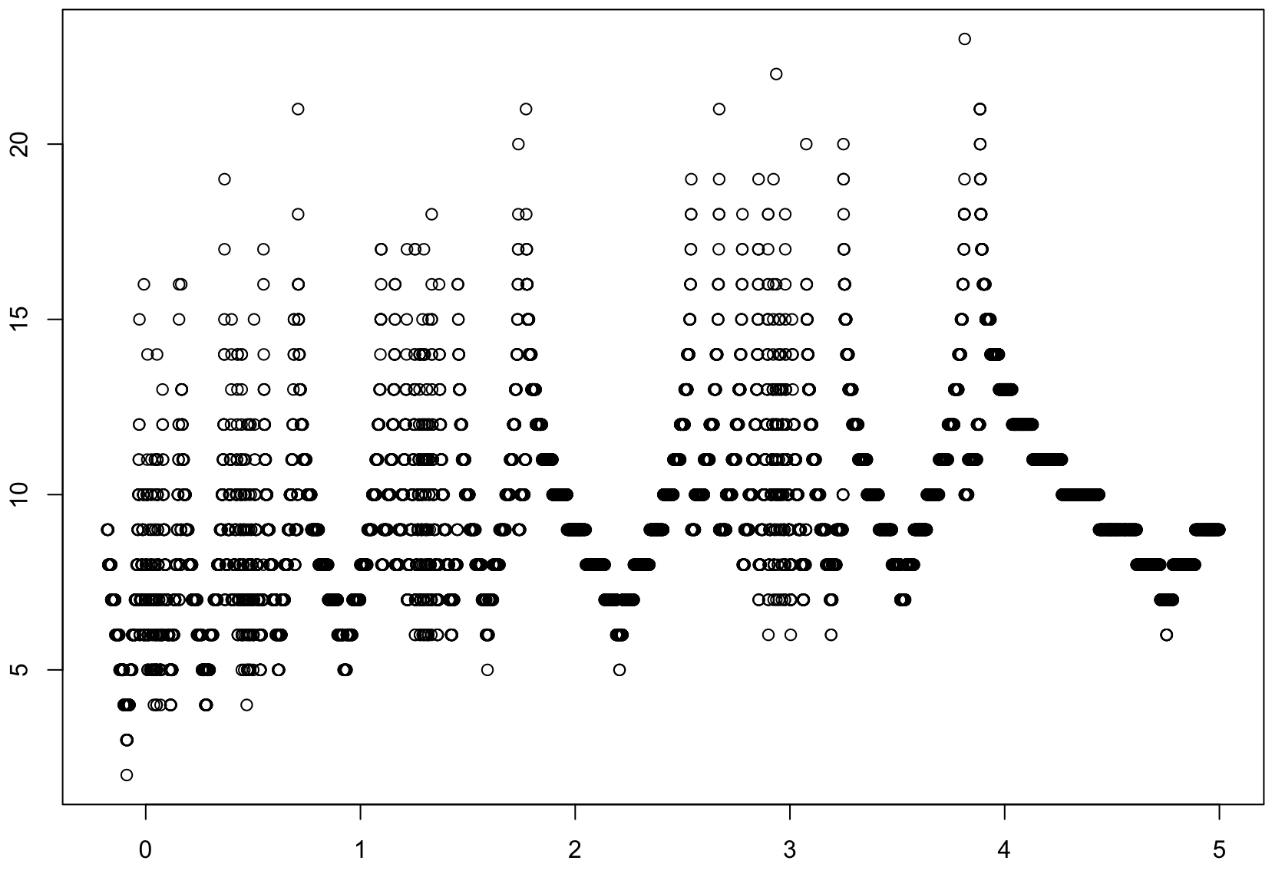

Fig.21 Initial value vs. output MLE based on 10000 different initial values ranging from -0.18 to 3000

We can see that the algorithm converges in no more than 23 iterations. The range of number of iterations for convergence is from 2 to 23, which is efficient.

In conclusion, Dyestuff2 dataset's larger residual variance makes the MLE of the NLL negative and with a smaller but steeper locally convex neighborhood. The smaller locally convex neighborhood makes the algorithm more sensitive to the initial value, since the reliable initial value interval for the vanilla version of Newton's method shrinks smaller. Also, since the convex neighborhood around the MLE is much steeper, the inverse Hessian doesn't give a large step size, which makes the convergence of the vanilla Newton's method slower, takes almost 3 times the number of iterations for the Dyestuff one. The vanilla Newton's method still fails on the right non-convex part of the NLL. After implementing the same algorithm, i.e., Newton's method with 1) direction correction through taking the absolute value of the Hessian, and 2) Halve the step size iteratively when the yielded parameter is invalid, the algorithm runs perfectly with any initial value between -0.18 and 5: It always converges to the reliable MLE robustly and fast within 23 iterations. The evidence is strong since it passes all the challenges over a big set of 10000 uniformly sampled initial values spanning from -0.18 to 5. Finally, we draw the conclusion that the MLE for the Dyestuff2 dataset is $\hat{\tau} \approx -0.08844658$. The result is much smaller than the one obtained with Dyestuff, but the optimization procedure does not change.