Homework 3

Mihai Pasnicu

October 10, 2018

1 Problem 1

The results are summarized in the table below.

2 Problem 2

I opted to use $fminunc$, which is basically Broyden. The results are once again summarized in the table.

3 Problem 3

It's still Broyden's method.

4 Problem 4

Table on the other page.

5 Note

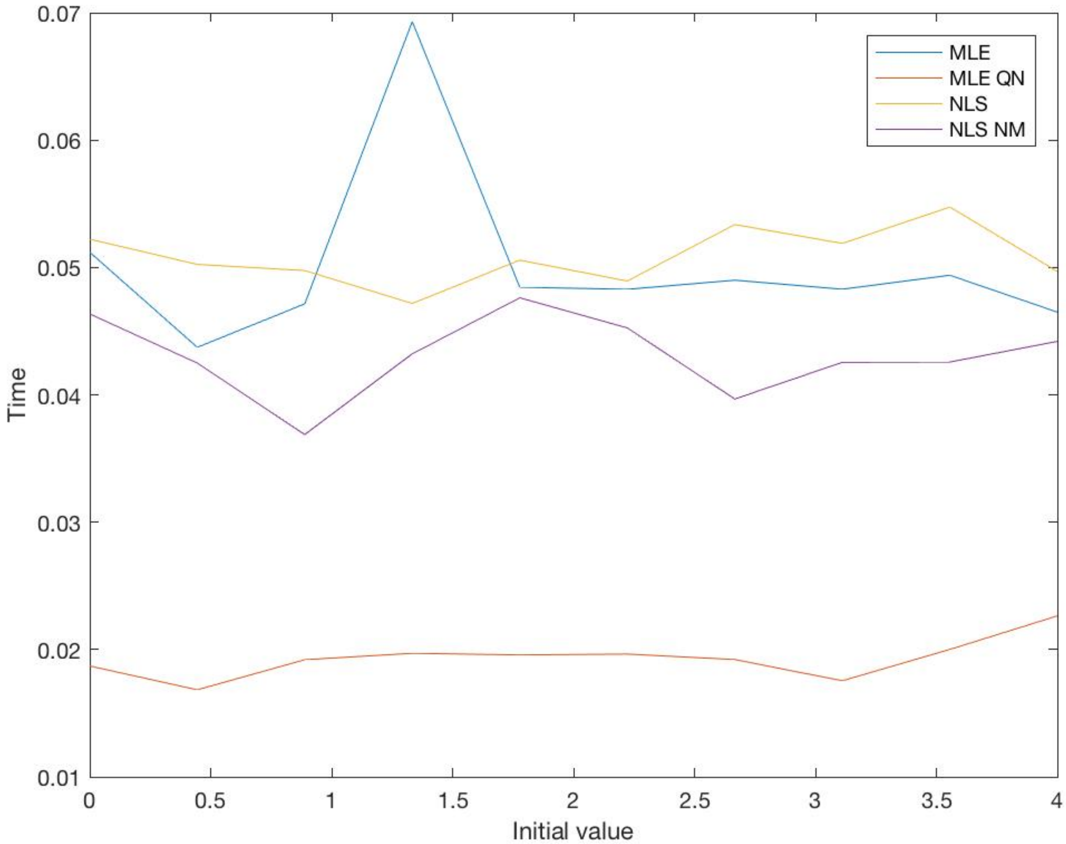
I don't really understand the last question in respect to the first 4. Since we were asked to estimate a parameter through NM, I opted to use a for loop in order to guarantee that both the MLE and NLS methods that use the Nelder - Mead simplex method will eventually reach the minimum and thus the correct solution. Therefore, in question number 5 it is to be expected that these methods will stay robust regardless of the starting value.

It is to be expected, and we checked, that by removing the loop, all methods based on Nelder - Mead will not be robust to the starting point. This is due to the fact that they will search for a local minimum rather than a global minimum.

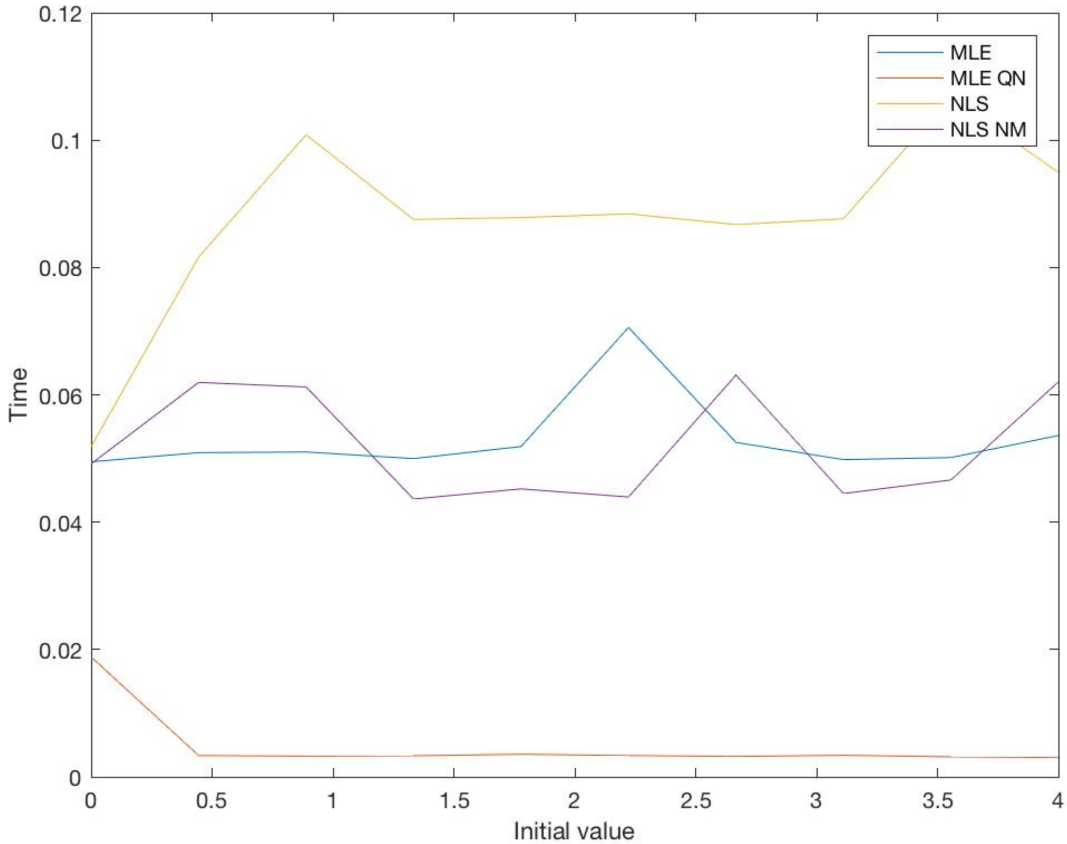
MLE	MLE_{QN}	NLS	NLS_{NM}
2.5339	2.5339	2.5126	2.5126
-0.0322	-0.0322	-0.0384	-0.0384
0.1157	0.1157	0.1141	0.1141
-0.3540	-0.3540	-0.2796	-0.2796
0.0798	0.0798	0.0676	0.0676
-0.4094	-0.4094	-0.3697	-0.3697

6 Observations problem 5

- I started with an initial value of $[1, 0, 0, 0, 0, 0]$ and then in order to check robustness, I looked across vectors of type $[k, 0, 0, 0, 0, 0]$, with k from 0 to 4, with k varying across all 6 positions.
- There is a huge difference in run times depending on the initial values. I have attached the graphs obtained by varying the first and the second beta, respectively.
- We can see that depending on the chosen method, MLE or NLS, we can choose a best approach. NM is definitely better when talking about NLS, but performs worst when looking at MLE and thus we cannot choose a universally best method.
- Lastly, I am going to reiterate the results in terms of robustness. The Nelder - Mead method is definitely way more sensitive to the initial value and it might not give the correct answer if not ran in a loop.



MLE
MLE QN
NLS
NLS NM
Time
Initial value
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0.5
1
1.5
2
2.5
3
3.5
4



Code:

```
function [t, b_sol] = hw3_code(b)

t = zeros(1 ,4);
b_sol = zeros(4 ,6);
M = load('hw3.mat');
x = M.X;
y = M.y;

%% problem 1

b1 = b;
f =@(b) - log_lik(x ,y ,b);

maxit = 100;
```

```
tic
for i = 1: maxit

    [b1, fval, exit] = fminsearch(f, b1, optimset('Display','final','TolFun',
    if (exit == 1)
        break
    end
end
t(1) = toc;
b_sol(1, :) = b1;


%% problem 2
b2 = b;
f2 = @(b) norm(log_lik(x, y, b));

tic
b2 = fminunc(f, b2, optimset('Display','final'));
t(2) = toc;
b_sol(2, :) = b2;


%% problem 3
b3 = b;
f3 = @(b)y-exp(x * b');

tic
b3 = lsqnonlin(f3, b3, [], [], optimoptions('lsqnonlin','Display','final','To
t(3) = toc;
b_sol(3, :) = b3;




%% problem 4
b4 = b;

f4 = @(b)sum((y - exp(x * b')) .^ 2);

tic
for i = 1: maxit

    [b4, fval, exit] = fminsearch(f4, b4, optimset('Display','final','TolFun'
    if (exit == 1)
        break
```

```
    end
end
t(4) = toc;
b_sol(4, :) = b4;
end
```