Assignment 1

02610 Optimization and Data Fitting – Anders Hørsted (s082382)

Question 1 - Branin's function

Branin's function $\mathbf{r} : \mathbb{R}^2 \to \mathbb{R}^2$ is defined as

$$\begin{aligned} r_1(\mathbf{x}) &= 1 - 2x_2 + \frac{1}{20}\sin(4\pi x_2) - x_1 \\ r_2(\mathbf{x}) &= x_2 - \frac{1}{2}\sin(2\pi x_1) \end{aligned} \tag{1}$$

and the function f is then defined as

$$f(\mathbf{x}) = \frac{1}{2}(r_1(\mathbf{x})^2 + r_2(\mathbf{x})^2) \tag{2}$$

Question 1.1

Since the function f is defined as the sum of two squared values we have that $f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^2$ and only for $r_1(\mathbf{x}) = r_2(\mathbf{x}) = 0$ is $f(\mathbf{x}) = 0$. From this it is seen that any solution of $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ is also a minimizer of f.

Question 1.2

A contour plot of $\log_{10}(f(\mathbf{x}))$ for $\mathbf{x} \in [-10; 10] \times [-10; 10]$ is plotted and shown in figure 1

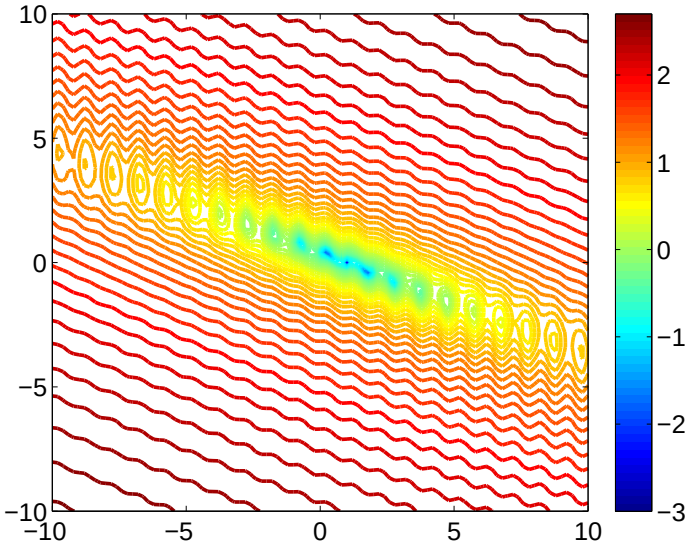

Figure 1: Contour plot of $\log_{10} f(\mathbf{x})$ as defined in question 1.

Question 2 - Newton's Method

Question 2.1

First an MATLAB implementation of Newton's method is created.

```
function [xmin,X,F,DF] = Newton(fun,x0,tol,maxiters,varargin)

    xmin = x0;
    k = 0;
    x = x0;

    [f,df,ddf] = feval(fun,x,varargin{:});

    X = [x0]; F = [f]; DF = [df];

    converged = (norm(df,'inf') < tol);

    while ~converged && k < maxiters
        s = -(ddf\df);
        x = x + s;

        [f,df,ddf] = feval(fun,x,varargin{:});

        X = [X,x]; F = [F,f]; DF = [DF,df];

        converged = (norm(df,'inf') < tol);

        if converged
            xmin = x;
        else
            xmin = [];
        end
```

```
        k = k + 1;
    end

end
```

Code Listing 1: Implementation of Newton's method

Question 2.2

The implementation is tested on the quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x}$ where

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -1.5 \\ 0 \end{pmatrix}$$

The extremum $\mathbf{x}^*$ can be found analytically by using $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, which gives

$$\mathbf{A}\mathbf{x}^* + \mathbf{b} = 0 \quad \Leftrightarrow \quad \mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{b} = \begin{pmatrix} 1 \\ -0.5 \end{pmatrix}$$

Using the implementation of Newton's method we find the same minimizer as the analytical solution, and Newton's method uses only one iteration. This is as expected since the first iteration $\mathbf{x}^{(1)}$ is given by (using that $\nabla^2 f(\mathbf{x}) = \mathbf{A}$)

$$\begin{aligned} \mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - \mathbf{A}^{-1}(\mathbf{A}\mathbf{x}^{(0)} + \mathbf{b}) \quad \Leftrightarrow \\ \mathbf{A}\mathbf{x}^{(1)} + \mathbf{A}\mathbf{x}^{(0)} + \mathbf{b} &= \mathbf{A}\mathbf{x}^{(0)} \quad \Leftrightarrow \\ \mathbf{x}^{(1)} &= -\mathbf{A}^{-1}\mathbf{b} \end{aligned}$$

so $\mathbf{x}^* = \mathbf{x}^{(1)}$ for the quadratic function f.

Question 2.3

Now the implementation of Newton's method is tested* on f – as defined in (2) – for four different starting guesses: $\mathbf{x}_1^{(0)} = (0,0)^T, \mathbf{x}_2^{(0)} = (1,0)^T, \mathbf{x}_3^{(0)} = (3.9,-1)^T, \mathbf{x}_4^{(0)} = (4.1,-1)^T$. The results are shown in figure 2. For the first three starting guesses the algorithm converges within the first 10 iterations giving the minimizers

$$\mathbf{x}_1^{(8)} = \begin{pmatrix} 0.1487 \\ 0.4021 \end{pmatrix}, \quad \mathbf{x}_2^{(0)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \mathbf{x}_3^{(6)} = \begin{pmatrix} 3.7305 \\ -1.2306 \end{pmatrix}$$

but for the starting guess $\mathbf{x}_4^{(0)}$ the algorithm do not converge within the allowed 1000 iterations. The reason for the lack of convergence can be found by inspecting the last few iterations. For the last 4 iterations the $\mathbf{x}$ values are

$$\mathbf{x}_4^{(997)} = \begin{pmatrix} 13.5082 \\ -2.3639 \end{pmatrix}, \quad \mathbf{x}_4^{(998)} = \begin{pmatrix} 13.5629 \\ -2.5779 \end{pmatrix}, \quad \mathbf{x}_4^{(999)} = \begin{pmatrix} 13.5082 \\ -2.3639 \end{pmatrix}, \quad \mathbf{x}_4^{(1000)} = \begin{pmatrix} 13.5629 \\ -2.5779 \end{pmatrix}$$

*Using a tolerance of 10^{-12} and a maximum of 1000 iterations.

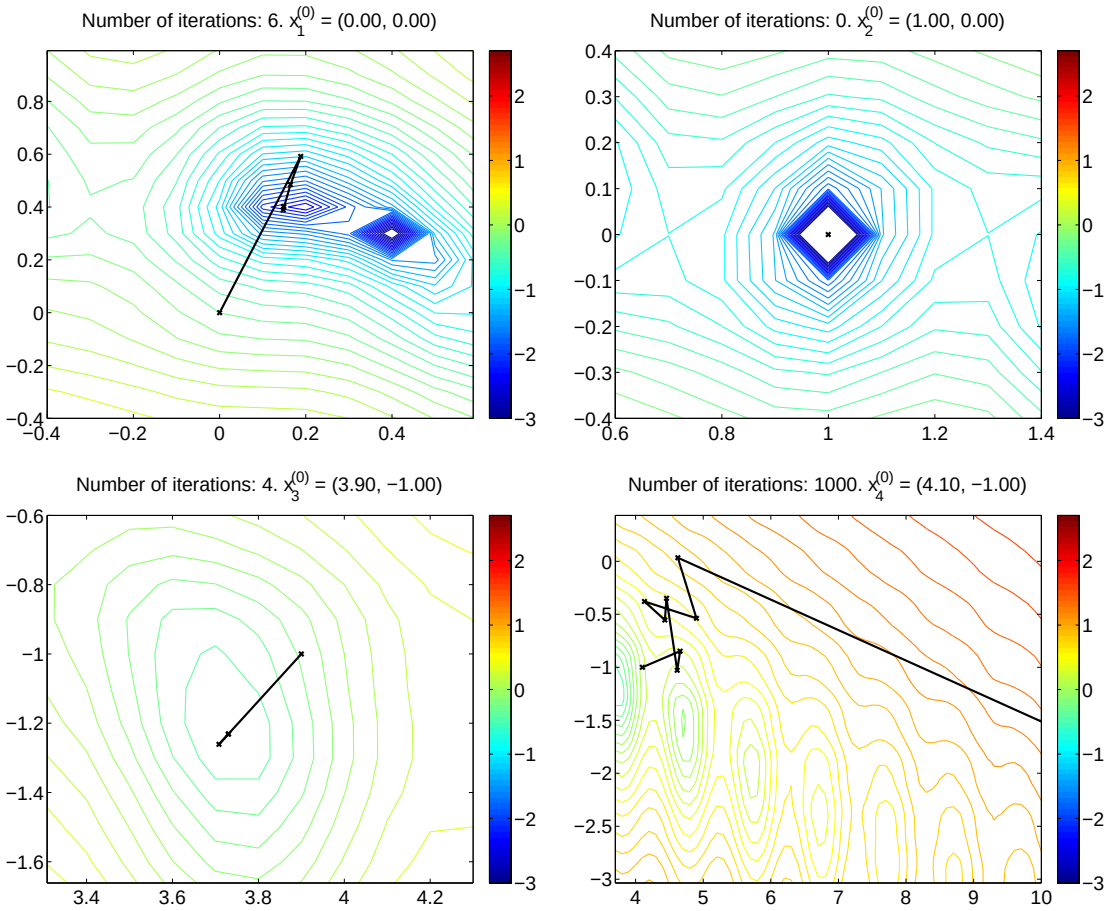

Figure 2: Testing implementation of Newton's method shown in code listing 1

and it do look like the algorithm is cycling between two different $\mathbf{x}$ values. Closer inspection of the $\mathbf{x}$ values shows that this is indeed the case.

It is now tested whether the found minimas are solutions for $\mathbf{r}(\mathbf{x}) = 0$ or only local minimizers for $f(\mathbf{x})$. For $\mathbf{x}_2^{(0)} = (1,0)^T$ it is seen from the expression (1) that it is a solution. For the other three candidates the norms of $\mathbf{r}$ are calculated as

$$||\,\mathbf{r}(\mathbf{x}_1^{(8)})\,||_2 = 2.78 \cdot 10^{-17}, \quad ||\,\mathbf{r}(\mathbf{x}_3^{(6)})\,||_2 = 7.86 \cdot 10^{-1}, \quad ||\,\mathbf{r}(\mathbf{x}_4^{(1000)})\,||_2 = 7.82$$

Which shows that $\mathbf{x}_1^{(8)}$ is a solution, but $\mathbf{x}_3^{(6)}$ and $\mathbf{x}_4^{(1000)}$ are ony local minimizers of f.

Question 3 - Least Squares Methods

In this question different function will be fitted to a dataset containing measurements of light intensity in a optical fibre as a function of time after source cut off.

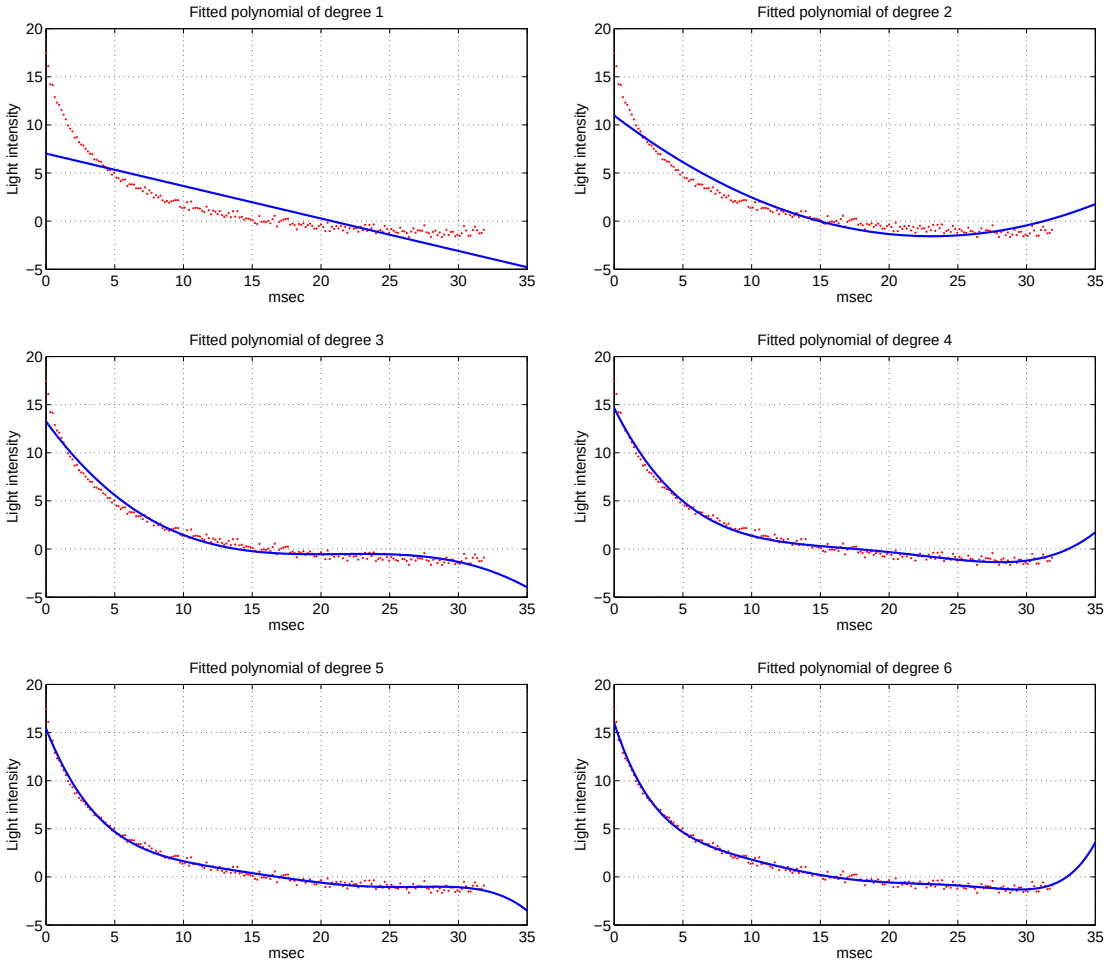

Figure 3: Fitted polynomials of degrees 1 to 6.

Question 3.1

First polynomials of degrees from 1 to 6 are fitted to the data using the MATLAB function `polyfit`. The results are shown in figure **??** and from the figure it is seen that the fit can be improved for values inside the data interval, by raising the degree of the polynomial. This is as expected since we could make the polynomial fit the data exactly by choosing the degree equal to the number of datapoints minus one (as long as there aren't two measurements for the same t). Outside the data interval though the behaviour of the fitted polynomials do not behave as well since all polynomials makes a sharp upward or downward turn when $t > 32$. This behaviour do not match the physical reality well, so instead of using polynomials, exponential functions are fitted instead.

Question 3.2

Now the data $\{(t_i, y_i)\}_{i=1}^m$ is fitted by the model

$$M(x,t) = x_2 e^{-x_1 t} + x_3 \tag{3}$$

First a function returning the residual vector and the jacobian matrix for the data given the parameters **x** as input is implemented. To make the function more generic we let it receive the data `t` and `y` as second and third argument. The actual data can then be passed in to the `marquardt` function.

```
function m1 = M1(x, t)
    m1 = x(2)*exp(-x(1)*t) + x(3);
end
```

Code Listing 2: Function returning value of model 1

```
function [r, J] = residual_jacobian_M1(x, t, y)

    r = M1(x,t) - y;
    J = [-x(2)*t.*exp(-x(1)*t) exp(-x(1)*t) ones(size(t))];

end
```

Code Listing 3: Function returning residual vector and jacobian for a given data set

The actual model is defined in a separate function `M1` since it is going to be reused when plotting the model. To ensure that the function is correctly implemented it is checked by running the `checkgrad` function.

```
[maxJ, err, ind] = checkgrad(@residual_jacobian_M1, x0, 1e-5, t, y);
```

The function returns the maximum value J_m of the calculated jacobian, the maximum error of the forward- (δ_F), backward- (δ_B), and extrapolated difference approximations and the indicies of where the maximum difference approximation errors was obtained. If the jacobian was correctly implemented it should be expected that $\delta^B \simeq \frac{1}{2}\delta^F$ and that δ^E should be of the order of magnitude $\simeq (\delta^F)^2$ REFERENCE!!!. The actual results was

$$\delta^F = 4.05 \cdot 10^{-5}, \quad \delta^B = -2.03 \cdot 10^{-5}, \quad \delta^E = -3.29 \cdot 10^{-10}$$

which strongly indicates that the `residual_jacobian_M1` function was correctly implemented.

Question 3.3

A reasonably starting point for the least squares fitting should now be obtained, from figure 2 in the assignment description. From (3) it is seen that for any x, $M(x,t) \to x_3$ for $t \to \infty$. From the figure a starting guess for x_3 could be $x_3 = -1$. From the figure is looks as if

$$M(x,0) = 17 \quad \Leftrightarrow \quad x_2 - 1 = 17 \quad \Leftrightarrow \quad x_2 = 18$$

Finally it is seen from the figure that approximately

$$M(x,15) = 0 \quad \Leftrightarrow \quad 18e^{-15x_1} - 1 = 0 \quad \Leftrightarrow \quad x_1 = \frac{\log(18)}{15} \approx 0.2$$

A decent starting guess will therefore be $\mathbf{x}^{(0)} = (0.2, 18, -1)^T$

Question 3.4

It is now time to fit the model (3) using the `marquardt` function from `immoptibox`. Using the start guess found in previous question the Levenberg-Marquardt method fits the model

$$M_1^*(t) = 15.66e^{-0.19t} - 0.95$$

to evaluate the model the fit is plotted along with the data. Also a plot of the residuals is created. Both plots can be seen in figure 4. From the plot of the fitted model the fit seems to be decent, but the model is seen to consistently give too high values for $t \in [2;6]$ and too low values for $t \in [6;15]$. This is confirmed by the residuals plot where it is also seen that the model fits really bad for t near 0. Also there seems to be some structure left in the residuals which shouldn't be there if the fit is perfect.

Comparing the model with the polynomial models found earlier, it is seen that the polynomial models of degree 5 and 6 fitted the data better inside the data interval ($t \in [0;32]$), but the exponential model behaves more plausible outside the data interval.

COMMENT ON THE PERFORMANCE OF L-M!!!

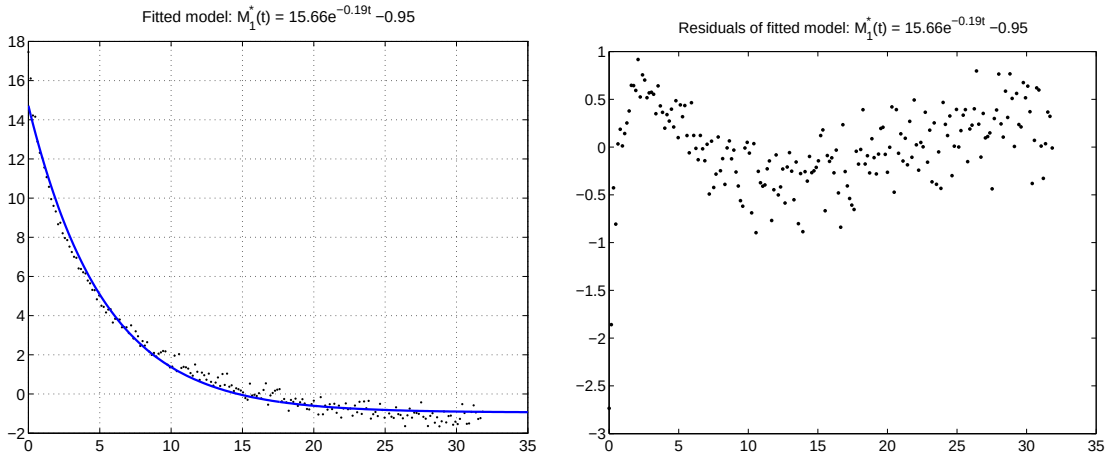

Figure 4: Plot of fitted model and residuals for model given by (3)

Iteration	$\|\| \nabla f \|\|$	Iteration	$\|\| \nabla f \|\|$
1	1.13e+03	1	1.16e+02
2	3.86e+01	2	6.57e+01
3	9.79e+01	3	3.98e+01
4	2.99e+01	4	2.46e+01
5	3.47e+01	5	1.99e+01
6	1.08e+01	6	1.96e+01
7	2.34e+00	7	1.95e+01
8	4.02e-01	8	1.95e+01
9	6.31e-02	9	1.95e+01
10	9.71e-03	10	1.95e+01
11	1.48e-03	11	1.95e+01
12	2.26e-04	12	1.95e+01
13	3.45e-05	13	1.95e+01
14	5.27e-06	14	1.95e+01
15	8.03e-07	15	1.95e+01
16	7.17e-08	16	1.95e+01

Question 3.5

Since the residuals of the previous model still showed some structure an extended model is now fitted. The model is given as

$$M_2(x,t) = x_3 e^{-x_1 t} + x_4 e^{-x_2 t} + x_5 \tag{4}$$

To fit the model a starting guess needs to be found. Using the fitted M_1 model a good starting point could be

$$\mathbf{x}^{(0)} = (0.2, 0.2, 8, 8, -1)^T$$

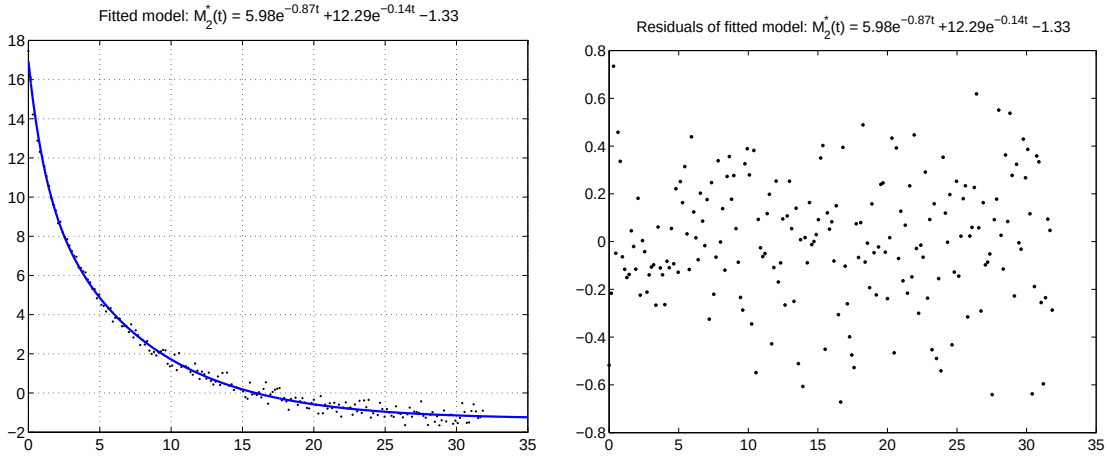

Figure 5: Plot of fitted model and residuals for model given by (4)

Using this starting guess a least squares fit for the new model is found using the `marquardt` function and the optimal model is found as

$$M_2^*(t) = 5.98e^{-0.87t} + 12.29e^{-0.14t} - 1.33$$

The new model is plotted along with the residuals of the model in figure 5

Question 3.6

From figure 5 it is seen that the fit has improved overall compared with model M_1. The residuals looks much more like white noise which is expected for a well-fitted model. It is often assumed that the variance of the residuals is independent of the input t, but it is seen that the variance is smaller for $t \in [1; 8]$ than for other t values. This could maybe be fixed with a transformation of the data, but shouldn't be a big problem. Most importantly it is seen that the residuals center around a mean of 0 for all values of t. All in all the fit seems to be satisfactory for the data.

Question 4

Question 4.1