Optimalisering og regulering TTK4135 - Assignment 9

Ola Kristoffer Hoff

27 April, 2023

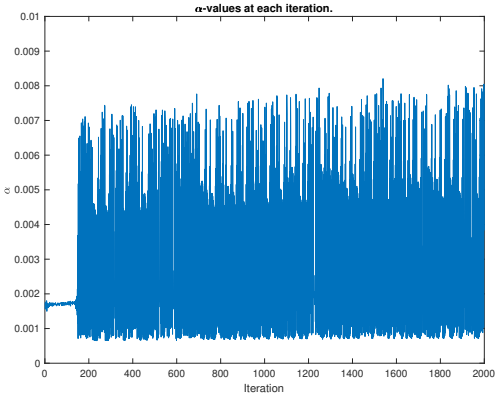
1 Unconstrained Optimization

This problem is about minimizing the Rosenbrock function

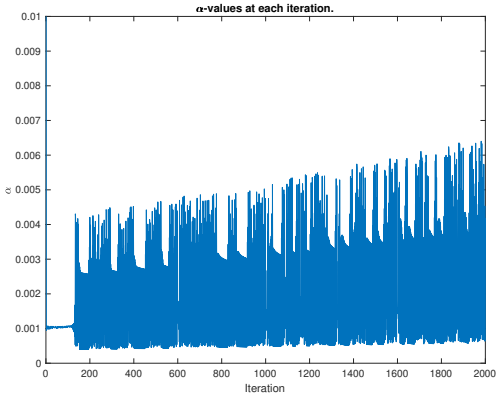
$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1}$$

a Solve Problem 3.1 in the textbook using MATLAB. Comment on your results. What does the condition in the loop of Algorithm 3.1 (page 37) ensure?

As we can tell from figure 1 and 2, Newton is a lot faster. The steepest descent did not converge to the minimum in the maximum iterations run. It stopped at $x_1^* = \begin{bmatrix}1.0086 & 1.0172\end{bmatrix}$ and $x_2^* = \begin{bmatrix}1.1421 & 1.3042\end{bmatrix}$, respectively. Whereas Newton found the minimum in 7 and 20 iterations, respectively.

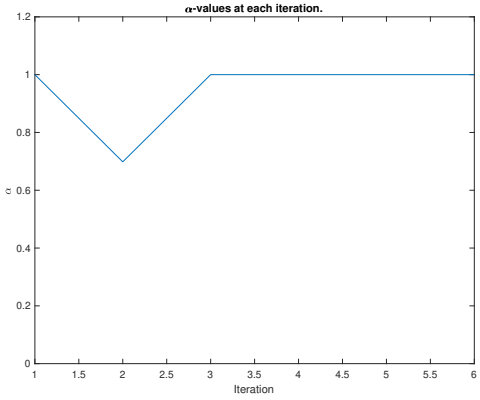


((a)) $x = \begin{bmatrix}1.2 & 1.2\end{bmatrix}$

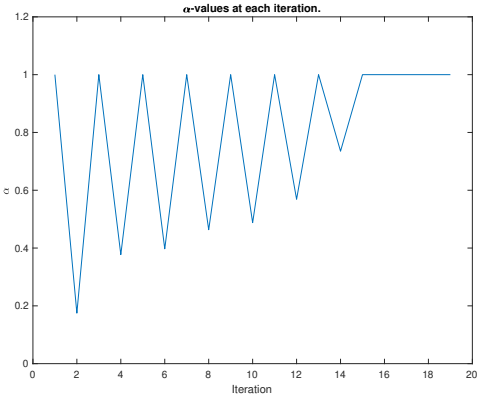


((b)) $x = \begin{bmatrix}-1.2 & 1\end{bmatrix}$

Figure 1: Plots of α-values used in iterations for the different start positions, with steepsest descent.



((a)) $x = \begin{bmatrix}1.2 & 1.2\end{bmatrix}$



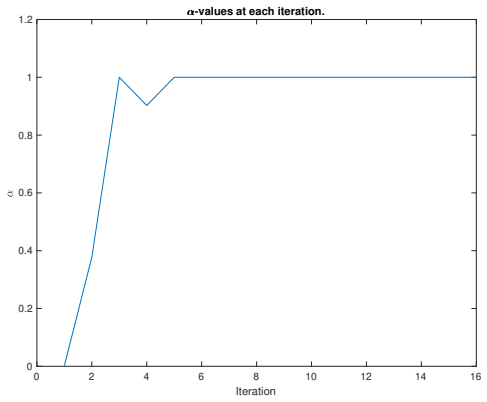
((b)) $x = \begin{bmatrix}-1.2 & 1\end{bmatrix}$

Figure 2: Plots of α-values used in iterations for the different start positions, with Newton.

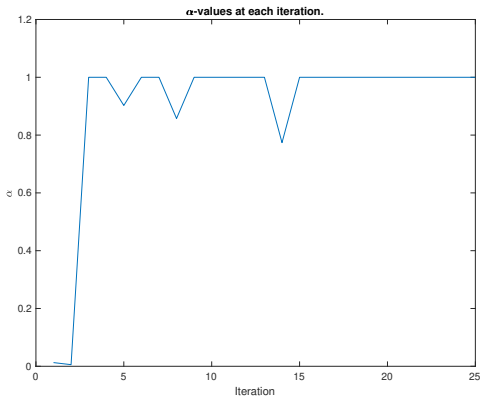
The algorithm 3.1 ensures that the selection of the α-value yields a sufficient decrease in the decent direction.

b Modify your code to use the BFGS method (the most common Quasi-Newton method), instead of using the exact Newton direction. Use Algorithm 6.1 (p. 140). Compare your results to the two algorithms developed in a) and comment.

This method seems to be even faster. It converged in 17 and 26 iterations respectively. It's as expected a bit slower than the exact Newton method. However, it is still very close to the same performance and vastly out perform the steepest descent method.



((a)) $x = \begin{bmatrix} 1.2 & 1.2 \end{bmatrix}$



((b)) $x = \begin{bmatrix} -1.2 & 1 \end{bmatrix}$

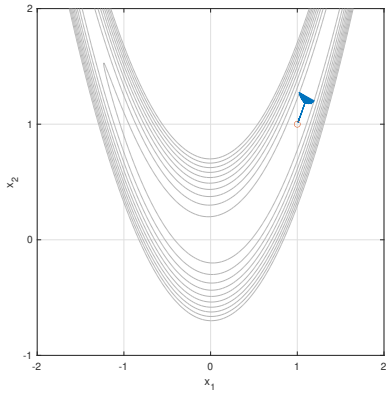
Figure 3: Plots of α-values used in iterations for the different start positions, with BFGS.

c Attach plots showing $f(x_k)$ as a function of k, α_k as a function of k, and a plot that shows how x_k moves in the x_1-x_2 plane relative to isocurves for f(x). You can use the MATLAB file plot_iter_rosenbrock.m published online for the isocurve plot. Also attach printouts of all three codes to your homework.

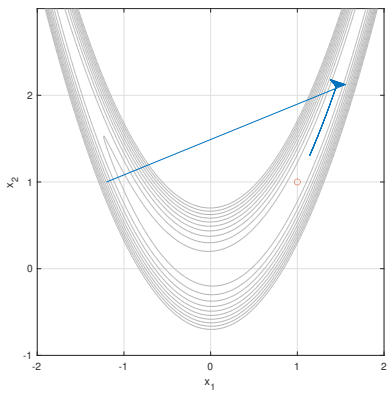
Study the plots that show α_k at each iteration k for the Newton and BFGS algorithms. How common is a step length of $\alpha_k = 1$? In particular, what is the step length close to the solution? How does this agree with the convergence-rate theory in Chapter 3.3?

Some of these plots were more natural to use in the previous subsections.

As discussed briefly above the α_k-values for BFGS and Newton were much better than steepest descent. The latter had no step lengths of 1, whereas the other two had many, a bit more for BFGS than Newton. This is probably due that it has more error to correct for so it can take bigger steps than Newton which is way more exact. The rate of step length equal to 1 seems to be greater towards the end, when nearing the solution. The number of iterations it takes the different methods correspond correctly to the theory, we see that they are linear, quadratic and super linear, respectively to steepest descent, Newton and BFGS.

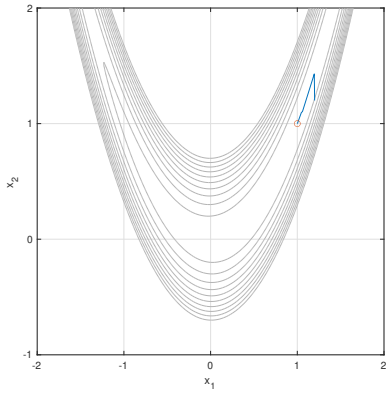


((a)) $x = \begin{bmatrix} 1.2 & 1.2 \end{bmatrix}$

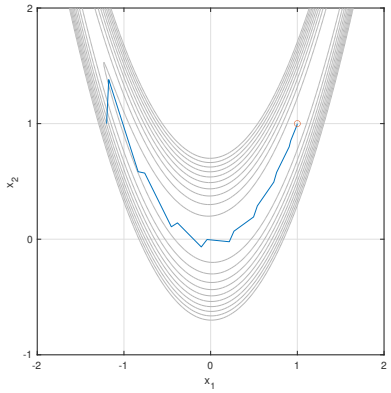


((b)) $x = \begin{bmatrix} -1.2 & 1 \end{bmatrix}$

Figure 4: Plots of $f(x_k)$ in iterations for the different start positions, with steepest descent.

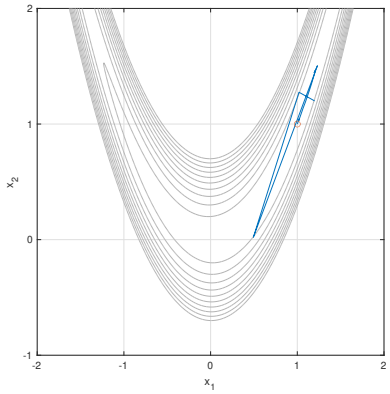


((a)) $x = \begin{bmatrix} 1.2 & 1.2 \end{bmatrix}$

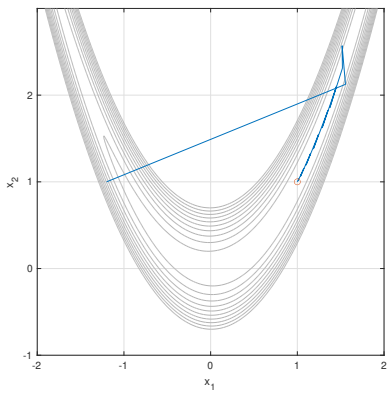


((b)) $x = \begin{bmatrix} -1.2 & 1 \end{bmatrix}$

Figure 5: Plots of $f(x_k)$ in iterations for the different start positions, with Newton.



((a)) $x = \begin{bmatrix} 1.2 & 1.2 \end{bmatrix}$



((b)) $x = \begin{bmatrix} -1.2 & 1 \end{bmatrix}$

Figure 6: Plots of $f(x_k)$ in iterations for the different start positions, with BFGS.

In plot 4(b) and 6(b) we have some interesting starting skips which I narrowed down to the α_0 value. This was sat to 1 as specified in the task. However, with a lower value the plots seemed more "correct". Then again, this was explicitly states in the task so I let it be.

2 Cholesky Factorization

The Newton direction can be modified using modified Cholesky factorization (p. 52 in the textbook). In which situations is it desirable to modify the search direction and what can be gained?

I the case where we have a Hessian that is not positive definite we can use Cholesky factorisation to gain a modified Hessian. Also the factorisation guarantees two properties: firstly, the Cholesky factors exist and are bound relative to the norm of the Hessian; secondly, it does not modify the Hessian if it is sufficiently positive definite[1].

3 Gradient Calculation

This problem is about calculation of gradients for

$$f(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2 \tag{2}$$

Note that this is not the Rosenbrock function. See Chapter 8 in the textbook for more on approximating derivatives.

a Derive an approximation of the gradient of $f(x)$ using the forward-difference scheme (see equation (8.1) in the textbook).

$$\begin{aligned}
\frac{\partial f}{\partial x_i}(x) &\approx \frac{f(x + \epsilon e_i) - f(x)}{\epsilon} \\
f(x) &= \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{f(x+\epsilon e_1)-f(x)}{\epsilon} \\ \frac{f(x+\epsilon e_2)-f(x)}{\epsilon} \end{bmatrix} \\
&= \begin{bmatrix} \frac{100(x_2-(x_1+\epsilon))^2+(1-(x_1+\epsilon))^2-100(x_2-x_1)^2-(1-x_1)^2}{\epsilon} \\ \frac{100((x_2+\epsilon)-x_1)^2+(1-x_1)^2-100(x_2-x_1)^2-(1-x_1)^2}{\epsilon} \end{bmatrix} \\
&= \begin{bmatrix} 200(x_1 - x_2) + 2(x_1 - 1) + 101\epsilon \\ 200(x_2 - x_1) + 100\epsilon \end{bmatrix}
\end{aligned}$$

b Calculate $\nabla f(x)$ analytically. Then, calculate $\nabla f(x)$ for $x = [0.5, 0.5]^T$ and $x = [1, 1]^T$ using three different values of ϵ for the approximation. Compare the approximations to the analytical gradient and discuss the results.

The analytic gradient would just be when ϵ approaches zero.

$$\begin{aligned}
&\lim_{\epsilon \to 0} \begin{bmatrix} 200(x_1 - x_2) + 2(x_1 - 1) + 101\epsilon \\ 200(x_2 - x_1) + 100\epsilon \end{bmatrix} \\
&= \begin{bmatrix} 200(x_1 - x_2) + 2(x_1 - 1) \\ 200(x_2 - x_1) \end{bmatrix}
\end{aligned}$$

In table 1 we see the results from the two x-values, with different ϵ values. As we can see as the ϵ converges on zero we approach the analytical result, as expected since that is how we found the analytic value. We can also see that for "big" values of ϵ (as 0.1) we get very incorrect answers.

x	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.001$	Analytical
$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 9.1 \\ 10 \end{bmatrix}$	$\begin{bmatrix} 0.01 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -0.899 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 10.1 \\ 10 \end{bmatrix}$	$\begin{bmatrix} 1.01 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.101 \\ 0.100 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Table 1: Comparison of approximation and analytical results for the gradient function.

c Discuss the error associated with the forward-difference approximation of $\nabla f(x)$.

As mentioned above, unless ϵ is sufficiently small, the error is quite big. Since the approximation function has a linear term for ϵ the error is also linear, as we can see from errors in table 1.

4 The Nelder-Mead Method

In this problem, the Rosenbrock function 1 will be minimized using the Nelder-Mead algorithm (a derivative-free method, see Section 9.5 in the textbook). To get a more intuitive idea of the Nelder-Mead method, this animation (https://secure.wikimedia.org/wikipedia/en/wiki/File:Nelder_Mead1.gif), showing the algorithm used on the Rosenbrock function, might be worth a look (the animation can be found on the Wikipedia page about the Nelder-Mead algorithm).

a The Nelder-Mead method needs $n + 1$ starting points for an n-dimensional function. What are the conditions on the $n+1$ initial points? Give an example of an invalid set of starting points. Describe in geometric terms the difference between valid and invalid sets of starting points for a function of two variables. Make sure you understand why this requirement is imposed.

So based on the illustration it seems like the method triangulates (in two dimensions) the position. Hence we need to form a "container" for the point in which means we need one more point than there is dimensions. These points obviously can't form linearly dependent vectors, that would in the 2D case mean that we had three points on a line (or a point). A set of identical points or points on a line (or a structure that can be formed by n points in n-dimensions) would be invalid, due to the linear dependence in the set.

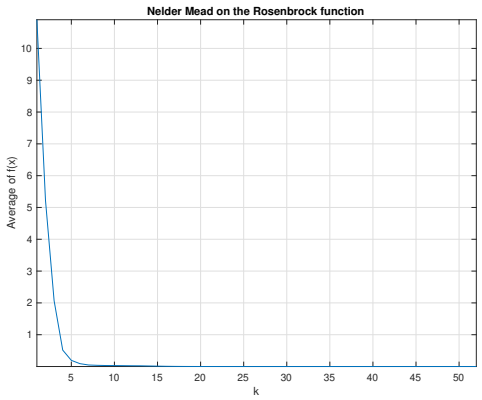
b Study the code *nelder_mead.m* posted on Blackboard and make sure you understand how it is related to Procedure 9.5 on page 238. Run the code with different initial points, look at the output and try different values for the parameters *fun_tol* and *x_tol*. Depending on the starting point you give the function, you may have to change the parameter iterlim as well. The code can be called like this:

$$x0 = [2, -2]'; \%\text{initial point}$$
$$[x, fval, x_iter] = nelder\_mead(x0,'report');$$

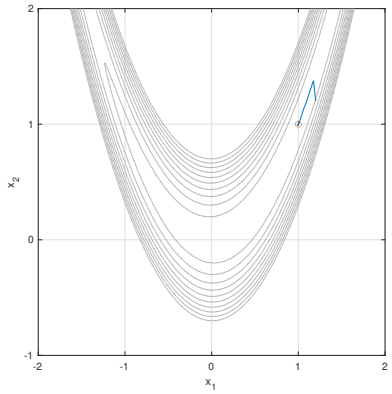
OK, done that.

c Use $x = [1.2, 1.2]^T$ as the initial point (this point is close to the optimum $x^* = [1, 1]^T$) and run the algorithm. Then start the algorithm from $x = [-1.2, 1]^T$, which is a more difficult starting point. Study the output and the plots. Discuss the shape of the curve illustrating the average value of $f(x)$ at each iteration. Plot the iterates in the x_1-x_2 plane relative to isocurves for $f(x)$. (Again, you can use the MATLAB file plot iter_rosenbrock.m published online.)

In figure 7 and 8 we can see the plots for the iterations and isocurves for $f(x)$. We can see that for the easier point it converges more quickly than the harder initial point. This is also evident from the curve plot where we can see that the better first starting point goes quite directly to the solution while the other one have to take a longer path, along the "banana".

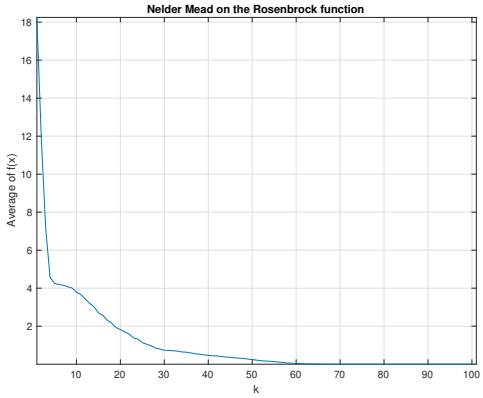


((a)) $f(x)$ through the iterations of the algorithm.

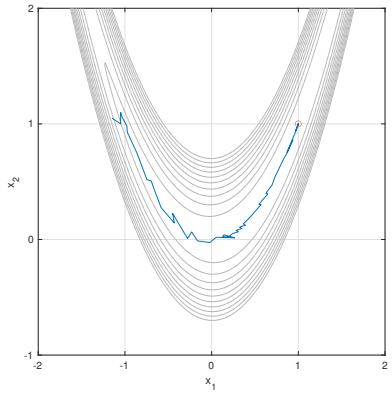


((b)) $x = \begin{bmatrix}1.2 & 1.2\end{bmatrix}$

Figure 7: Plots of $f(x_k)$ in iterations for the $x = \begin{bmatrix}1.2 & 1.2\end{bmatrix}$.



((a)) $f(x)$ through the iterations of the algorithm.



((b)) $x = \begin{bmatrix}1.2 & 1\end{bmatrix}$

Figure 8: Plots of $f(x_k)$ in iterations for the $x = \begin{bmatrix}1.2 & 1\end{bmatrix}$.

d Compare your results from c) with Problem 1. (You can measure the time an algorithm uses in MATLAB using the commands tic and toc, but this is not the most relevant means of comparison.)

In table 2 we see the number of iterations the different algorithms from task 1, and the Nelder-Mead method used on the two starting positions.

x	SD	Newton	BFGS	NM
$\begin{bmatrix}1.2\\1.2\end{bmatrix}$	DNF	7	17	52
$\begin{bmatrix}-1.2\\1\end{bmatrix}$	DNF	20	26	101

Table 2: Comparison of iterations used by the different algorithms on the Rosenbrock function.

e Solve the first half of Problem 9.12 on page 244 in the textbook. (That is, ignore the last sentence of the problem.)

The average value of the function over the simplex vertices is:

$$\frac{1}{n+1}\sum_{i=1}^{n+1} f(x_i)$$

The definition of a convex function is:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \quad \forall \alpha \in [0,1]$$

In the step of shrinking the simplex all $x_i, i > 1$ is sat to $\frac{1}{2}(x_1 + x_i)$. We can rewrite this into the convexity definition:

$$f(\frac{1}{2}x_1 + \frac{1}{2}x_i) \leq \frac{1}{2}f(x_1) + \frac{1}{2}f(x_i), \quad \forall i > 1 \tag{3}$$

We note that the points are in ascending order of value: $f(x_j) \leq f(x_i), \ \forall i > j$. Thus we get the following relation:

$$\frac{1}{2}f(x_1) + \frac{1}{2}f(x_i) \leq \frac{1}{2}f(x_i) + \frac{1}{2}f(x_i) = f(x_i), \qquad \forall i > 1 \tag{4}$$

Notice that the left hand side of 4 is the same as the right hand side of 3. Meaning we can substitute it for the right hand side in 4 while maintaining the relation constraint:

$$f(\frac{1}{2}x_1 + \frac{1}{2}x_i) \leq f(x_i), \qquad \forall i > 1$$

Now we can see that the average sum uses the term on the right hand side which is greater or equal to the one on the left hand side and by definition the left hand side is the new points after the shrinking meaning that we get the sums:

$$f_k = \frac{1}{n+1}\sum_{i=1}^{n+1} f(x_i) = \frac{1}{n+1}f(x_1) + \frac{1}{n+1}\sum_{i=2}^{n+1} f(x_i)$$
$$f_{k+1} = \frac{1}{n+1}\sum_{i=1}^{n+1} f(x_i) = \frac{1}{n+1}f(x_1) + \frac{1}{n+1}\sum_{i=2}^{n+1} f(\frac{1}{2}(x_1 + x_i))$$
$$f_{k+1} \leq f_k$$

Hence we use the last relation to see that the sum after the shrinking that the average value cannot increase.

References

[1] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Science+Business Media, LLC, 2006.