Numerical Optimization - Handin 4

Martin Simon Haugaard - CDL966

March 4, 2016

4.1

The Minimum can be determined by examining the fact that a circle with radius Δ has the following property:

$$\Delta^2 = x^2 + y^2$$

Which can be reformulated into

$$y = \sqrt{(}\Delta^2 - x^2) \tag{1}$$

Now calculating the Gradient and the Hessian for the given formula for any point, x.

$$\nabla f(x) = \begin{bmatrix} -40x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 20(x_2 - x_1^2) \end{bmatrix}$$
$$\nabla^2 f(x) = \begin{bmatrix} 40(x_2 - 3x_1^2) + 2 & -40x_1 \\ -40x_1 & 20 \end{bmatrix}$$

Calculating these for $x_1 = 0, x_2 = -1$

$$\nabla f(x) = 11$$
$$\nabla^2 f(x) = \begin{bmatrix} 42 & 0 \\ 0 & 20 \end{bmatrix}$$

All of these informations can then be placed into the formula $m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k^p$ which yields the following function

$$m_k(p) = 11 - 2x - 20y + 21x^2 + 10y^2$$

Now substituting in (1) the final formula becomes:

$$m_k(p) = 11x^2 - 2x + 11 - 20 * \sqrt{\Delta^2 - x^2} + 10\Delta^2$$

Using MATLAB's *fminsearch* function, the minimums of the final formula can be determined. And for various Δ radius from $0 < \Delta \leq 2$ Figure 1 is given:

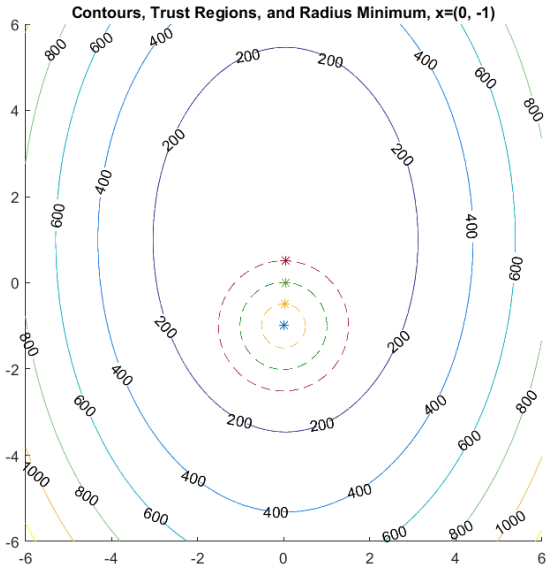

Figure 1: Contour Lines and Solutions for point $x = (0, -1)$.

The minimum values seem to make sense, as they all are as close to the minimum of the contour drawing as they can be, while remaining on the trust region border.

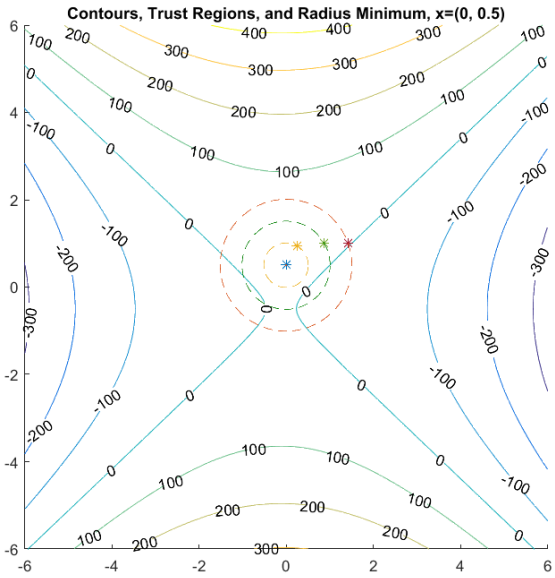

Figure 2: Contour Lines and Solutions for point $x = (0, 0.5)$.

Following the same steps as with $x = (0, -1)$ the contours for $x = (0, 0.5)$ are drawn and once again the (Figure 2).

4.10

B is symmetric, that means there can be an orthogonal matrix Q and a diagonal matrix $\Lambda = \{\lambda_1, \lambda_2, \ldots \lambda_n\}$, where $\lambda_1 \ldots \lambda_n$ are eigenvalues of B for which

$$B = Q\Lambda Q^T$$

If the lowest of the eigenvalues in Λ is greater than or equal to zero, then all the rest of the eigenvalues are also greater than or equal to zero, and then B is positive definite. Which means $B + \lambda I$ is also positive definite, since $0 \leq \lambda$.

If the lowest value eigenvalue is negative, then it's possible to choose a λ such that it is slightly greater than the absolute value of the lowest eigenvalue.

Now observing $B = Q\Lambda Q^T$. Doing the calculation $B + \lambda I$ is the same as

$$B + \lambda I = Q(\Lambda + \lambda I)Q^T$$

Which ensures that the diagonal of $\lambda_1 \ldots \lambda_n$ all become positive, since the absolute value of the lowest λ in the diagonal is being added to all of them. Since they are now all positive, the result is positive definite.

Programming Exercise - 4.2

Implementing the dogleg method, I start with a random Δ value, lower than the $\hat{\Delta}$, which is the maximum step length. I do the first iteration calculations with this random Δ value, calculating a τ. Once the τ is calculated, I calculate the trajectory as described in (4.16). Which I can then use in (4.3) to calculate the next step. Once I have the next step, I calculate the ratio ρ (4.4), which allows me to scale Δ as described in **Algorithm 4.1**. I then update my x and start a new iteration. I terminate the loop once the gradient for the x is sufficiently small, or after a certain (high) amount of iterations.

I plot my iterations as shows in Figure 3 & Figure 4, where I have also plotted the steepest descent method and also newton in Figure 3 for comparison.

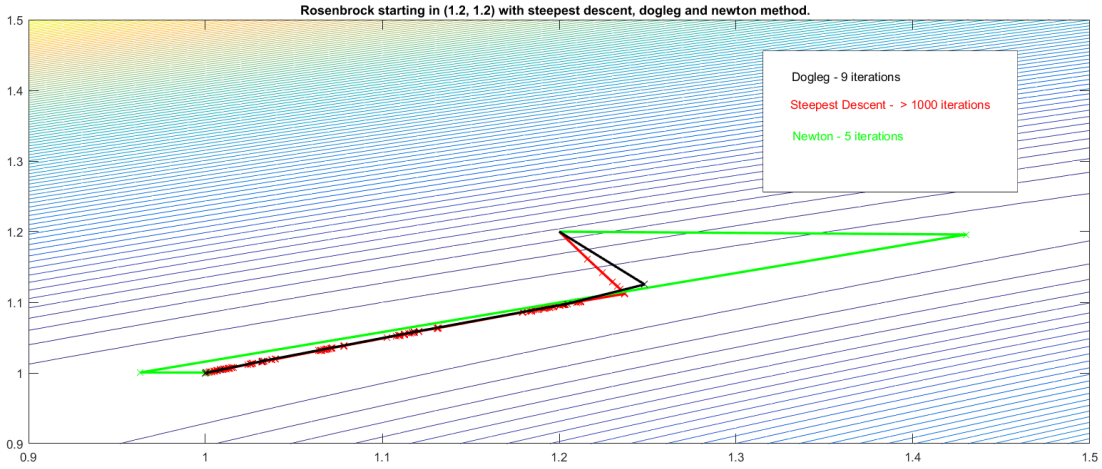

Figure 3: Steepest Descent, Newton and Dogleg method, starting at $(1.2, 1.2)$

As noticed last week, the steepest descent will never actually reach the minimum, and it will spend all of it's allowed iterations trying to approach it. Both the Newton and the the dogleg method reaches it fairly quickly however, after 5 and 9 iterations respectively. It's worth noting that the dogleg's approach seems a lot more smooth compared to the newton, as it's almost on top of the steepest descent.

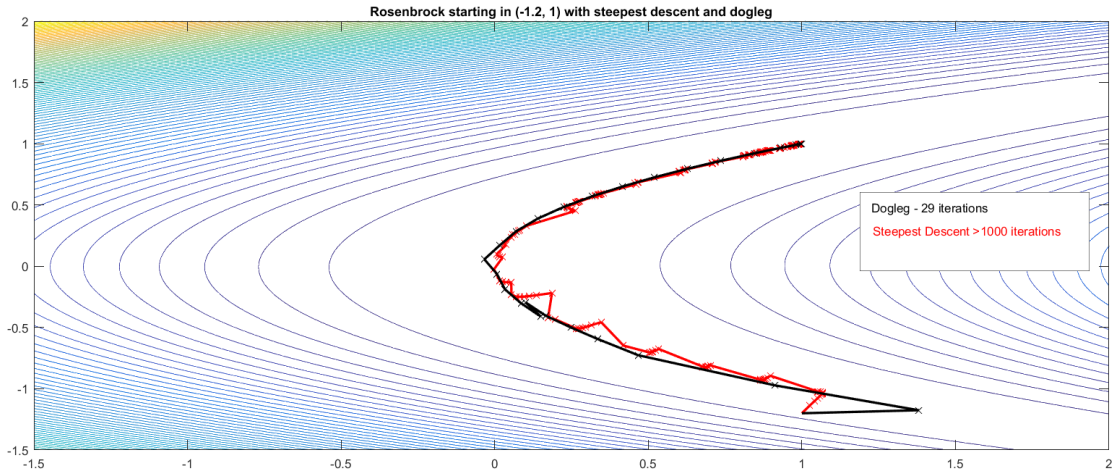

Figure 4: Steepest Descent and Dogleg method, starting at $(-1.2, 1)$

Last week I was unable to make my newton method find the minimum when starting in $x = (-1.2, 1)$ hence it's not included in Figure 4. Once again both the steepest descent and the dogleg approach the minimum, however the steepest descent never gets close enough for a satisfactory gradient. The dogleg does however, after only 29 iterations.

I can conclude that the dogleg method, even with it's slightly more expensive calculations, find the minimum of the rosenbrock function much faster than the steepest descent. I was slightly surprised by the fact that the newton method was four iterations faster than the dogleg in the $x = (1.2, 1.2)$ run. But it seems to make sense, as the descent needed to find the minimum is fairly obvious for $x = (1.2, 1.2)$, hence calculating the dogleg may be too much effort.