Numerical Algorithms for Visual Computing III 2011
Example Solutions for Assignment 4

Problem 1 (Comparison of Gradient Domain methods)

(a) The gradient vector $\nabla E_f(u)$ is

$$\frac{\partial E_f(u)}{\partial u_1} = (u_1 - f_1) - \alpha \frac{u_2 - u_1}{\Delta x^2} \quad (1)$$

$$\frac{\partial E_f(u)}{\partial u_i} = (u_i - f_i) - \alpha \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} \qquad i = 2 \dots N-1 \quad (2)$$

$$\frac{\partial E_f(u)}{\partial u_1} = (u_N - f_N) + \alpha \frac{u_N - u_{N-1}}{\Delta x^2} \quad (3)$$

$$(4)$$

and the Hessian $HE_f(u)$ is given by

$$HE_f(u) = \begin{pmatrix} 1 + \frac{\alpha}{\Delta x^2} & -\frac{\alpha}{\Delta x^2} & & 0 \\ -\frac{\alpha}{\Delta x^2} & 1 + 2\frac{\alpha}{\Delta x^2} & -\frac{\alpha}{\Delta x^2} & \\ \ddots & \ddots & \ddots & \\ & -\frac{\alpha}{\Delta x^2} & 1 + 2\frac{\alpha}{\Delta x^2} & -\frac{\alpha}{\Delta x^2} \\ 0 & & -\frac{\alpha}{\Delta x^2} & 1 + \frac{\alpha}{\Delta x^2} \end{pmatrix} \quad (5)$$

(b) A naive way to implement a stopping criterion is to check $\|u^{(k+1)} - u^{(k)}\|_2 < \varepsilon$ or $\|f(x^{(k)}) - p^*\| < \varepsilon$ for a fixed point p^*. This may be a good choice, however it may happen, that the algorithm stays in a local minimum. However, a good choice for a stopping criterion for this part of the exercise, as well as the next, would be to use the criterion $\|\nabla E_f(u^{k+1}) - \nabla E_f(u^k)\|_2 < \varepsilon$. With this choice, the algorithm stops at that time, if the gradient direction does no longer change. However it is also possible to check $\|\nabla E_f(u^{k+1})\|_2 < \varepsilon$, i.e. if the gradient direction is the zero vector. As the Hessian of the problem has only positive eigenvalues (not shown here), we can say that the function is convex and thus, the found minimum is also the local and global minimum.

(c) The Newton method finds also the global minimum but with less iterations compared to the GDLS algorithm. This is due to the fact that the use of the inverted Hessian in the search direction does employ a gradient descent direction in the Hessian norm

$$\|u\|_{Hf(x)} = (u^\top Hf(x)u)^{\frac{1}{2}}. \quad (6)$$

As the Hessian matrix is constant for a fixed α, the Hessian directly helps finding a descent direction to the minimum.

Problem 2 (Playing with the banana) The Rosenbrock function is a polynomial of 4th degree and has a global minimum in $x = (1, 1)^\top$. It is a non-convex function where the minimum lies in a banana-like, parabolic-shaped valley along the parabola $x_2 = x_1^2$. The other test function is also a polynomial of 4th degree with a global minimum at $(2, 1)^\top$. The problem that occurs with this function is that the Hessian is singular at this minimum, which may lead to some problems for algorithms that depend on positiv definite Hessians. In the following, we will always employ the stopping criterion $\|\nabla f\| < 0.000001$. For comparison, the gradients are

$$\nabla f_{Banana} = \begin{pmatrix} -2(1-x) - 400x * (y - x^2) \\ 200y - 200x^2 \end{pmatrix} \tag{7}$$

$$\nabla f_{B-S} = \begin{pmatrix} 4(x-2)^3 + 2x - 4y \\ -4x + 8y \end{pmatrix} \tag{8}$$

and the Hessians

$$Hf_{Banana} = \begin{pmatrix} 2 + 1200x^2 - 400y & -400x \\ -400x & 200 \end{pmatrix} \tag{9}$$

$$Hf_{B-S} = \begin{pmatrix} 12(x-2)^2 + 2 & -4 \\ -4 & 8 \end{pmatrix} \tag{10}$$

1. For the Banana function, the Newton scheme converges after 6 iterations to the correct value $(1, 1)^\top$. For the Bazaraa-Shetty function however, the scheme converges after 13 iterations to a value of $(1.9835577, 0.99117788)^\top$, which is close, but still relatively far away from the correct solution, due to the singular Hessian for the minimum.

2. For the Banana function, a choice of $\alpha = 0.01$ and $\beta = 0.25$ leads to the approximated solution $(1.0000586, 1.0001175)^\top$ after 150 iterations. In this exercise, the choice of α and β (of the line search algorithm) can be substantial here, as it may lead to divisions by zero. Therefore, one should also make sure that $d^\top q$ should be bigger than zero. For example, for $\alpha = 0.1, \beta = 0.3$ the algorithm converges with 9430 iterations. The same settings however let the algorithm converge for a result of the Bazaara-Shetty function to a result $(2.0259357, 1.0129725)^\top$ after 430 iterations. With a setting of $\alpha = 0.3$ and $\beta = 0.11$ it converges after 8 iterations to $(1.9733595, 0.9866750)^\top$.

3. Now the trust region method should be used for cases, in which indefinite or singular Hessians perturb the outcome of the algorithm. The idea of the trust region method is to shift the eigenvalues into a positive range. There are several possible ideas how to do that, we will show two. A first method is to compute all eigenvalues of a matrix and determine the minimal eigenvalue. In case this one lies in the negative range, the α should be chosen in such a way, that $\lambda_{\min} + \alpha > 0$, for example by choice of $|\lambda_{\min}| + 1$. Another idea which does not need to compute the eigenvalues directly depends on a diagonally dominance. By Gerschgorin's Theorem, a strictly diagonally dominant matrix has only positive eigenvalues. In that sense, one has to find the row that "violates" the diagonally dominance the most and add an α in such a way that the Hessian is again diagonally dominant. In this case, we employ a standard Newton method as a basis iteration scheme. For the starting point $(-1.2, 1)$, the scheme converges after 6 iterations, which we have already seen in part (a). The reason for that is, from this point on, the Hessian is always positive definite. If one chooses e.g. as a starting point $(-1.2, 2)^\top$, the first Hessian contains negative eigenvalues, thus the Trust Region method can be used here. For the Bazaraa function, one can see that every point on the line $(2, y)^\top$ gives a singular matrix. Applied on such a point (e.g. $(2, 0)^\top$) converges after 8 iterations with $(1.9787172, 0.9893586)^\top$ as a result.

Problem 3 (Descending Into The Abyss)

(a) Given the fact, that $f'(x) = 2x$, a gradient descend scheme looks as follows:

$$x^{(k+1)} = x^{(k)} - 2 \cdot x^{(k)} = -x^{(k)} \tag{11}$$

and for the next iteration

$$x^{(k+2)} = x^{(k+1)} - 2 \cdot x^{(k+1)} = -x^{(k+1)} = x^{(k)}. \tag{12}$$

In this case, the gradient descent algorithm ends up in an infinite loop, in which the current search positions are landing directly opposite.

(b) The first derivative of the function is $f'(x) = 4x^3 + 9x^2 - 6x - 7$ and $f''(x) = 12x^2 + 18x - 6$. A curve sketching shows that local extrema are 1 and $-\frac{13}{8} \pm \frac{\sqrt{57}}{8}$. However, the function has 2 minima and 1 maxima, i.e.

it is not a convex function. The global minimum however is located at $-\frac{13}{8} - \frac{\sqrt{57}}{8}$ Remember, that the Newton method for optimisation problems is a variant of the Newton method to find zero crossings, which is what is happening here. Extrema are zerocrossings of the first derivative, so the Newton method will find a zero crossing for a given starting point. However the choice of a starting point also determines in which extrema the result will end up with, so for example if the algorithm starts at $x = 5$, the Newton method will find the local minimum at position 1, also if you start with $x = 0$, then one ends up in the local maximum at $-\frac{13}{8} - \frac{\sqrt{57}}{8}$.

(c) The first derivative of $f(x)$ is $f'(x) = \frac{1}{x} - a$ and $f''(x) = -\frac{1}{x^2}$, i.e. the function does not have a minimum but a maximum and is concave. The Newton method is given by

$$x^{(k+1)} = x^{(k)} - \frac{\frac{1}{x^{(k)}} - a}{-\frac{1}{(x^{(k)})^2}} = 2x^{(k)} - a(x^{(k)})^2. \tag{13}$$

The result (i.e. the global minimum) of this algorithm is $\frac{1}{a}$, so this is actually a neat way to program an iterative algorithm for reciprocal values without any use of divisions.

However, this problem is not defined on $\mathbb{R}$ but on $\mathbb{R}^+$, so it may happen, that one search direction lands in an area where the original function is not defined. In case, one search direction would lead to a negative value, then the eventual result would be $-\infty$. So you have to choose your starting point relatively close to the desired result.

(d) Remember that the original Newton method tries to find zero crossings of a given function. Only when considering the first and the second derivative, the Newton method tries to find a minimum of a function. In this case, we analyse the simple method for $g(x) := f'(x) = \arctan(x)$. One can see that for starting values below a value $x \approx 1.39$ the method converges to the desired point. If a value above 1.39 is chosen, then the algorithm diverges. Also, it is possible to let the algorithm run in an infinite loop if some value ≈ 1.39 is chosen.

Problem 4 (Watch your step)
We have implemented the Armijo conditions by means of the Gradient Descent Line Search Algorithm for exercise 1. It is possible to reduce the number of iterations for large α values. However one must give several new parameters in advance. For example, with $\alpha = 0.1$ (as in the second exercise) and

$\beta_1 = 0.002$ and $\beta_2 = 0.5$ it is possible to find a solution after 205 iterations, instead of 9430 iterations. (In the actual implementation, the choice of finding a suitable $\beta \in [\beta_1\sigma, \beta_2\sigma]$ was realised by taking a linear combination $\alpha\beta_1\sigma + (1-\alpha)\beta_2\sigma$ with a given parameter λ).