Assignment 9

Problem 1: Unconstraint optimization

$$f(x) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$$

a) Use algorithm 3.1 with $\alpha_0 = 1$.
Try the two initial points:

$$x_0 = [1.2,\ 1.2]^T$$

$$x_0 = [-1.2,\ 1]^T$$

Comment on results:

From the results on can see that Newton's algorithm in general takes larger steps than the steepest descent algorithm, and is thus able to find the optimal point in a smaller number of iterations.

The condition in the loop of algorithm (3.1) ensures the Wolfe condition meaning that each step of the algorithm gives a sufficient decrease in the objective function.

b) From the plots one can see that the BFSG algorithm in general takes larger steps than both Newton's algorithm and the steepest descent algorithm. This makes the algorithm converge faster when the gradients are not that large, as is the case with $x_0 = [1.2, 1.2]^T$. It does however make the algorithm use more iterations when the gradients are large. This due to the algorithm overshooting, as can be seen in the case where $x_0 = [-1.2, 1.0]^T$.

c) A step length $\alpha = 1$ is more common with the BFGS than with Newton, even though it is quite frequent with both algorithms. From the plots one can see that $\alpha = 1$ is more frequent closer to the optimal solution.

This agrees with theorem 3.5 and the Wolfe condition for Newton's algorithm and with theorem 3.6 for the BFGS algorithm.

Problem 2: Cholesky factorization

Newton direction:

$$p_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k$$

Since $\nabla^2 f_k$ might not be positive definite, its inverse $(\nabla^2 f_k)^{-1}$ might also not be positive definite. In this case the Newton direction is not a descent direction.

By using modified Cholesky factorization one can replace $(\nabla^2 f_k)^{-1}$ with a positive definite approximation, making the Newton direction a descent direction.

Problem 3 : Gradient calculation

$$f(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2$$

a)

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right]^T$$

Using forward-difference scheme to approximate the partial derivatives:

$$\frac{\partial f}{\partial x_1} \approx \frac{f(x + \Delta x_1) - f(x)}{\Delta x_1}$$

$$= \Big(100(x_2 - (x_1 + \Delta x_1))^2 + (1 - (x_1 + \Delta x_1))^2$$

$$- 100(x_2 - x_1)^2 - (1 - x_1)^2 \Big)$$

$$/ \Delta x_1$$

$$= \Big(100(\cancel{x_2^2} - 2x_2(\cancel{x_1} + \Delta x_1) + (x_1 + \Delta x_1)^2)$$

$$+ \cancel{1} - 2(\cancel{x_1} + \Delta x_1) + (x_1 + \Delta x_1)^2$$

$$- 100(\cancel{x_2^2} - \cancel{2x_2}x_1 + x_1^2)$$

$$- \cancel{1} + \cancel{2}x_1 - x_1^2 \Big) / \Delta x_1$$

$$= \Big(100 \left(-2x_2 \Delta x_1 + \cancel{x_1^2} + 2x_1 \Delta x_1 + \Delta x_1^2 \right)$$
$$- 2\Delta x_1 + \cancel{x_1^2} + 2x_1 \Delta x_1 + \Delta x_1^2$$
$$\cancel{-100 (x_1^2)}$$
$$\cancel{- x_1^2} \Big) / \Delta x_1$$

$$= \big(-200 x_2 \Delta x_1 + 200 x_1 \Delta x_1 + 100 \Delta x_1^2$$
$$- 2\Delta x_1 + 2x_1 \Delta x_1 + \Delta x_1^2 \big) / \Delta x_1$$

$$= \frac{\left(-200 x_2 \Delta x_1 + 202 x_1 \Delta x_1 + 101 \Delta x_1^2 \right)}{\Delta x_1} - 2$$

$$= -200 x_2 + 202 x_1 + 101 \Delta x_1 - 2$$

$$\frac{\partial f}{\partial x_1} \approx \frac{f(x + \Delta x_2) - f(x)}{\Delta x_2}$$

$$= \Big(100 \left((x_2 + \Delta x_2) - x_1 \right)^2$$
$$\cancel{+ (1 - x_1)^2}$$
$$- 100 (x_2 - x_1)^2$$
$$\cancel{- (1 - x_1)^2}$$
$$\Big) / \Delta x_2$$

$$= \Big(100 \big((x_2 + \Delta x_2)^2 - 2(x_2 + \Delta x_2)x_1 + x_1^2 \big) - 100 \big(x_2^2 - 2x_1x_2 + x_1^2 \big) \Big) / \Delta x_2$$

$$= \Big(100 \big(x_2^2 + 2x_2\Delta x_2 + \Delta x_2^2 - 2x_1\Delta x_2 \big) - 100 \big(x_2^2 \big) \Big) / \Delta x_2$$

$$= 100 \big(2\Delta x_2 (x_2 - x_1) + \Delta x_2^2 \big) / \Delta x_2$$

$$= 100 \big(2(x_2 - x_1) + \Delta x_2 \big)$$

$$= 200 x_2 - 200 x_1 + 100 \Delta x_2$$

Setting $\Delta x_1 = \Delta x_2 = \varepsilon$

$$\Longrightarrow \quad \nabla f(x) \approx \begin{bmatrix} 202x_1 - 202x_2 - 2 + 101\varepsilon \\ -200x_1 + 200x_2 + 100\varepsilon \end{bmatrix}$$

b) $f(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2$

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}\left(100(x_2 - x_1)^2 + (1 - x_1)^2\right)$$

$$= -200(x_2 - x_1) - 2(1 - x_1)$$

$$= -200x_2 + 200x_1 - 2 + 2x_1$$

$$= 202x_1 - 200x_2 - 2$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}\left(100(x_2 - x_1)^2 + (1 - x_1)^2\right)$$

$$= 200(x_2 - x_1)$$

$$= 200x_2 - 200x_1$$

$$\nabla f(x) = \begin{bmatrix} 202x_1 - 200x_2 - 2 \\ -200x_1 + 200x_2 \end{bmatrix}$$

$x = [0.5 \quad 0.5]^T$

Anal. grad.	Num. $\varepsilon = 10^{-2}$	Num. $\varepsilon = 10^{-4}$	Num. $\varepsilon = 10^{-6}$
$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.01 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -0.9899 \\ 0.01 \end{bmatrix}$	$\begin{bmatrix} -0.9999 \\ 0.0001 \end{bmatrix}$

$x = [0 \quad 0]^T$

Anal. grad	Num. $\varepsilon = 10^{-2}$	Num. $\varepsilon = 10^{-4}$	Num. $\varepsilon = 10^{-6}$
$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1.0100 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.0101 \\ 0.0100 \end{bmatrix}$	$\begin{bmatrix} 0.0001 \\ 0.0001 \end{bmatrix}$

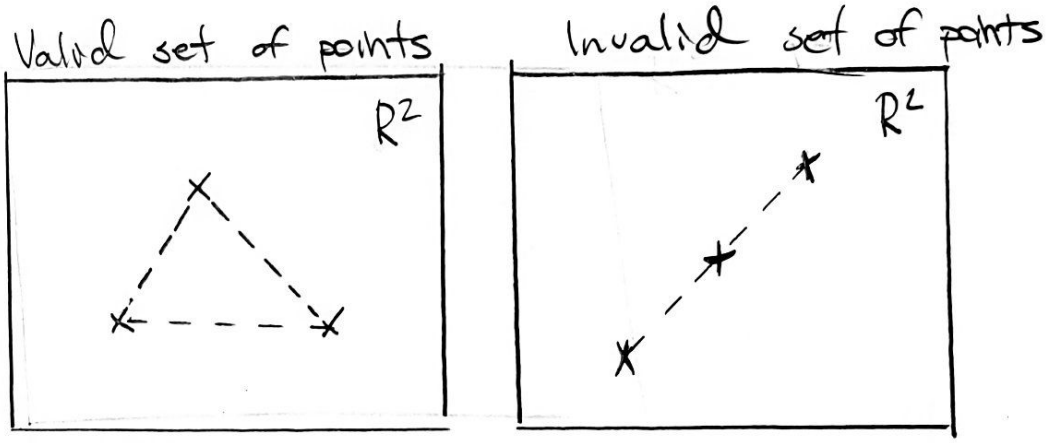
c) The forward scheme approximation of the gradient is only a first order approximation of the gradient and can thus produce wildly wrong approximations of $\nabla f(x)$ if ε is not sufficiently small.

Problem 4: The Nelder-Mead method

a) The requirement of the Nelder-Mead method is that the matrix associated with the simplex $S = \{z_1, z_2, \ldots, z_{n+1}\}$,

$$V(S) = [z_2 - z_1, z_3 - z_1, \ldots, z_{n+1} - z_1]$$

is nonsingular. This is achieved if the vertices $z_k - z_i$ are not coplanar.



b)

c) From the plots one see that the Nelder-Mead algorithm monotonically decrease the average function value when starting at $x_0 = [1.2, 1.2]^T$. The algorithm finds the optimal solution in about 7 iterations. When starting at $x_0 = [-1.2, 1]^T$ one can see that the algorithm stagnates after about

7 iterations. After the stagnation the average function value decreases very slowly before the algorithm finds the optimal solution in about 60 iterations.

This is consistent with experiments with the Nelder-Mead algorithm, which suggests that it has a tendency of stagnating at certain points.

d) When not stagnating the Nelder-Mead algorithm converges faster than the SD, Newton and BFGS algorithm. However when stagnating the algorithm is on par with Newtons and BFGS.

e) f is a convex function

$$\updownarrow$$

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$$

Shrinkage step:

$$x_{k+1}^i = \frac{1}{2}(x_k^1 + x_k^i), \quad i = 2, \ldots, n+1$$

$$\alpha = \frac{1}{2} \implies f\left(\frac{1}{2}x + \frac{1}{2}y\right) \leq \frac{1}{2}f(x) + \frac{1}{2}f(y)$$

Setting $x = x_k^1$ and $y = x_k^i$:

$$f\left(\frac{1}{2}x_k^1 + \frac{1}{2}x_k^i\right) \leq \frac{1}{2}f(x_k^1) + \frac{1}{2}f(x_k^i)$$

$$\overline{f}_k = \frac{1}{n+1}\sum_{i=1}^{n+1} f(x_k^i)$$

$$\overline{f}_{k+1} = \frac{1}{n+1}\sum_{i=1}^{n+1} f(x_{k+1}^i) = \frac{1}{n+1}\sum_{i=1}^{n+1} f\left(\frac{1}{2}x_k^1 + \frac{1}{2}x_k^i\right)$$

$$= \frac{1}{n+1}\left(f(x_k^1) + \sum_{i=2}^{n+1} f\left(\frac{1}{2}x_k^1 + \frac{1}{2}x_k^i\right)\right)$$

$$\overline{f}_{k+1} = \frac{1}{n+1} f(x_k^1) + \frac{1}{n+1} \sum_{i=2}^{n+1} f\left(\frac{1}{2}x_k^1 + \frac{1}{2}x_k^i\right)$$

$$\overline{f}_k = \frac{1}{n+1} \sum_{i=1}^{n+1} f(x_k^i)$$

$$\overline{f}_{k+1} = \frac{1}{n+1} f(x_k^1) + \frac{1}{n+1} \sum_{i=2}^{n+1} f\left(\frac{1}{2}x_k^1 + \frac{1}{2}x_k^i\right)$$

$$\leq \frac{1}{n+1} f(x_k^1) + \frac{1}{n+1} \sum_{i=2}^{n+1} \left(\frac{1}{2}f(x_k^1) + \frac{1}{2}f(x_k^i)\right)$$

$$\overline{f}_{k+1} \leq \frac{1}{n+1} f(x_k^1) + \frac{1}{n+1} \cdot \frac{n}{2} f(x_k^1)$$

$$+ \frac{1}{2} \cdot \frac{1}{n+1} \sum_{i=2}^{n+1} f(x_k^i)$$

$$\leq \frac{n+2}{2(n+1)} f(x_k^1) + \frac{1}{2} \frac{1}{n+1} \sum_{i=2}^{n+1} f(x_k^i)$$

$$\leq f(x_k^1) + \frac{1}{n+1} \sum_{i=2}^{n+1} f(x_k^i)$$