Assignment 1

Callum Lau, 19102521

Numerical Optimisation

January 30, 2020

1 PSD Matrix

1.1 (a)

Note that A is clearly symmetric since $A^\top = (B^\top B)^\top = B^\top B = A$. Furthermore note that if v is an eigenvector of A with eigenvalue λ then

$$\begin{aligned} Av = \lambda v &\implies B^\top Bv = \lambda v \\ &\implies v^\top B^\top Bv = \lambda v^\top v \\ &\implies ||Bv||^2 = \lambda ||v||^2 \\ &\implies \lambda = \frac{||Bv||^2}{||v||^2} \implies \lambda \geq 0 \end{aligned}$$

Now a matrix is positive semi-definite if and only if its eigenvalues are greater than or equal to zero. This is precisely what we have shown above and therefore A is a symmetric positive semi-definite matrix.

1.2 (b)

A function $f(x)$ is convex $\iff f(y + \alpha(x-y)) - \alpha f(x) - (1-\alpha)f(y) \leq 0$. We show this inequality holds:

$$\begin{aligned} &f(y + \alpha(x-y)) - \alpha f(x) - (1-\alpha)f(y) \\ &\quad = (y + \alpha(x-y))^\top A(y + \alpha(x-y)) - \alpha x^\top Ax - (1-\alpha)y^\top Ay) \\ &\quad = y^\top Ay + \alpha y^\top A(x-y) + \alpha(x-y)^\top Ay + \alpha^2(x-y)^\top A(x-y) - \alpha x^\top Ax - y^\top Ay + \alpha y^\top Ay \\ &\quad = \alpha(y^\top Ax + x^\top Ay - y^\top Ay - x^\top Ax) - \alpha^2(y^\top Ax + x^\top Ay - y^\top Ay - x^\top Ax) \\ &\quad = (\alpha - \alpha^2)(y^\top Ax + x^\top Ay - y^\top Ay - x^\top Ax) \\ &\quad = (\alpha - \alpha^2)(y-x)^\top A(y-x) = (\alpha - \alpha^2)z^\top Az \leq 0 \end{aligned}$$

where the last inequality holds since $\alpha \in [0,1] \implies (\alpha - \alpha^2) \leq 0$ and A is symmetric positive semidefinite $\implies z^\top Az \geq 0$ for all vectors $z \in \mathbb{R}^n$. Therefore we have $f(y + \alpha(x-y)) - \alpha f(x) - (1-\alpha)f(y) \leq 0$ and hence $f(x)$ is convex.

2 Taylor Expansion

2.1 (a)

Firstly we note that:

$$\begin{aligned} f(x) &= \cos(1/x) \\ f'(x) &= \frac{1}{x^2}\sin(1/x) \end{aligned}$$

$$f''(x) = -\frac{2}{x^3}\sin(1/x) - \frac{1}{x^4}\cos(1/x)$$

Also we have that the general form of a 2nd order Taylor expansion at $x \in \mathbb{R}$ with $a \in \mathbb{R}$ small, can be written

$$f(x+a) \approx f(x) + f'(x)a + \frac{1}{2}f''(x)a^2$$

And therefore in the case $f(x) = \cos(1/x)$:

$$f(x+a) \approx \cos(1/x) + \frac{1}{x^2}\sin(1/x)a - \left[\frac{1}{x^3}\sin(1/x) + \frac{1}{2x^4}\cos(1/x)\right]a^2$$

2.2 (b)

$$\begin{aligned} g(\boldsymbol{x}) &= \exp(||\boldsymbol{x}||^2) \\ \nabla g(\boldsymbol{x}) &= 2\boldsymbol{x}\exp(||\boldsymbol{x}||^2) \\ \nabla^2 g(\boldsymbol{x}) &= \begin{pmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 g}{\partial x_2^2} \end{pmatrix} = 2\exp(||\boldsymbol{x}||^2)\begin{pmatrix} 1+2x_1^2 & 2x_1x_2 \\ 2x_1x_2 & 1+2x_1^2 \end{pmatrix} \end{aligned}$$

Now note that the general of a 2nd order Taylor expansion at $\boldsymbol{x} \in \mathbb{R}^2$ with small $\alpha \in \mathbb{R}^+$ in direction $\boldsymbol{p} \in \mathbb{R}^2$:

$$g(\boldsymbol{x}+\alpha\boldsymbol{p}) \approx g(\boldsymbol{x}) + \alpha\boldsymbol{p}^\top\nabla g(\boldsymbol{x}) + \frac{1}{2}\alpha^2\boldsymbol{p}^\top\nabla^2 g(\boldsymbol{x})\boldsymbol{p}$$

And therefore in the case $g(\boldsymbol{x}) = \exp(||\boldsymbol{x}||^2)$:

$$\begin{aligned} g(\boldsymbol{x}+\alpha\boldsymbol{p}) &\approx \exp(||\boldsymbol{x}||^2)\left(1 + 2\alpha\boldsymbol{p}^\top\boldsymbol{x} + \alpha^2\boldsymbol{p}^\top\begin{pmatrix} 1+2x_1^2 & 2x_1x_2 \\ 2x_1x_2 & 1+2x_1^2 \end{pmatrix}\boldsymbol{p}\right) \\ &= \exp(||\boldsymbol{x}||^2)\left(1 + 2\alpha(p_1x_1 + p_2x_2) + \alpha^2\left(p_1^2(1+2x_1^2) + 4p_1p_2x_1x_2 + p_2^2(1+2x_2^2)\right)\right) \end{aligned}$$

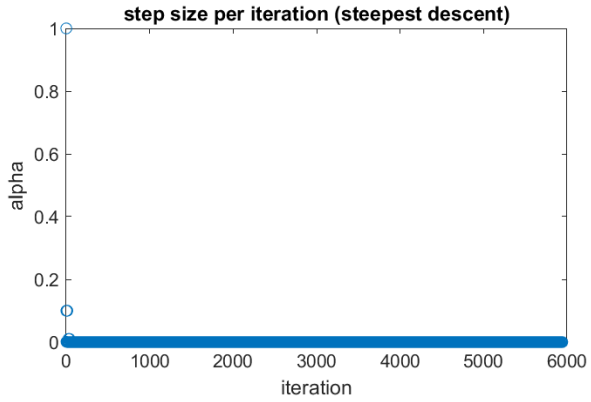
3 Backtracking, Steepest Descent and Newton's algorithms

3.1 (a)

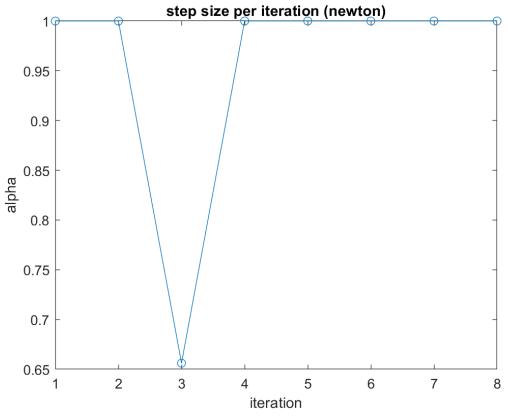
Answers submitted via `MATLAB` *Grader*.

3.2 (b)

For the steepest descent and Newton algorithm, starting at $x_0 = (1.2, 1.2)^T$ with initial step length $\alpha_0 = 1$, the plot of step size is given below:

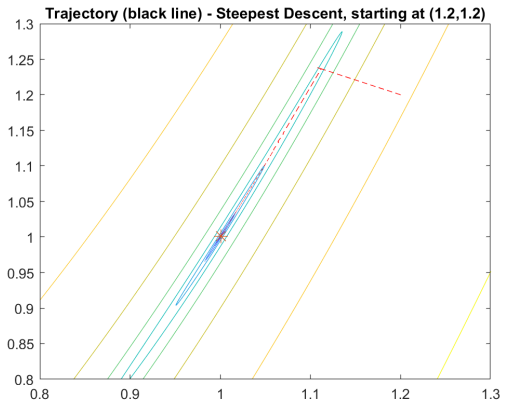


(a) Steepest Descent

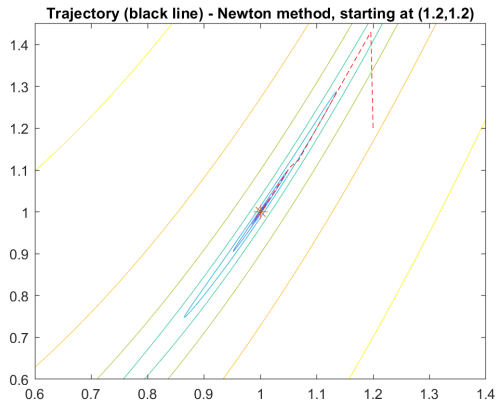


(b) Newton's

The plot of the trajectories of x_k against the contours of the Rosenbrock function for the steepest descent and Newton algorithms is given below:



(a) Steepest Descent



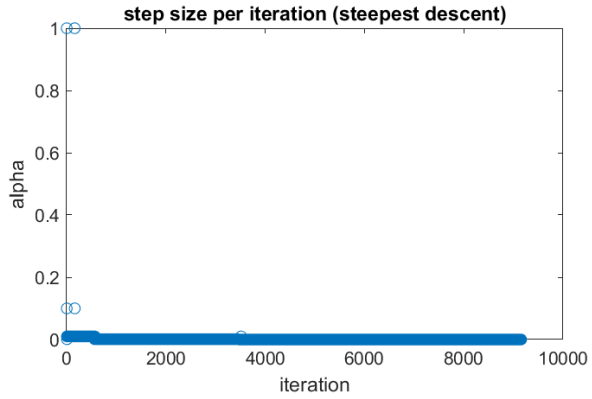
(b) Newton's

The trajectories (dotted red line) can be explained as follows: the steepest descent algorithm will ensure that the descent direction $\boldsymbol{p}$ to follow in the minimisation procedure is $\boldsymbol{p} = -\nabla f(\boldsymbol{x})$, hence why the trajectory crosses the contours at a 90^o degree angle (not shown here exactly due to scaling). It then changes direction sharply towards the minimum $(1,1)^T$(bottom left point) and follows the 'saddle point' between two contour 'hills'. Note that as it approaches the minimum $(1,1)^T$ it requires a significantly more of iterations (the dotted red lines get closer and closer together as it approaches $(1,1)^T$).

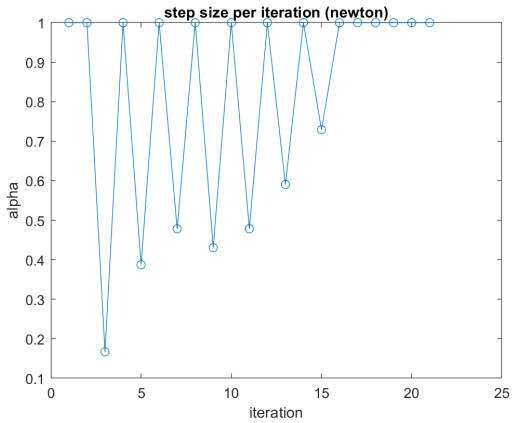
Conversely Newton's algorithm uses both first and second order derivatives in the descent direction and therefore does not have to cross the contours perpendicular. Instead the Newton direction is set to minimise the second order Taylor polynomial of the Rosenbrock function which may explain why it changes direction to follow the saddle point without overstepping. Note also that it requires far fewer iterations to reach the minimum.

3.3 (c)

We repeat the calculations from the starting point $(-1.2,1)^T$. The step sizes are shown below:

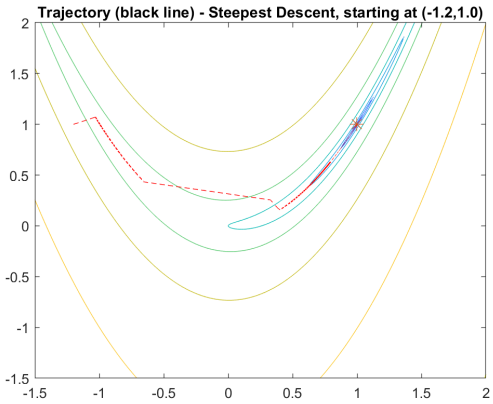


(a) Steepest Descent

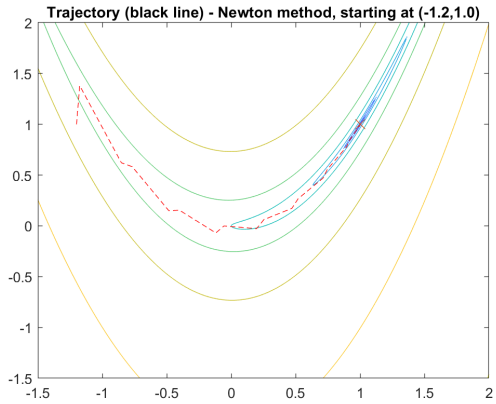


(b) Newton's

Additionally, the trajectories for both algorithms are also given:



(a) Steepest Descent

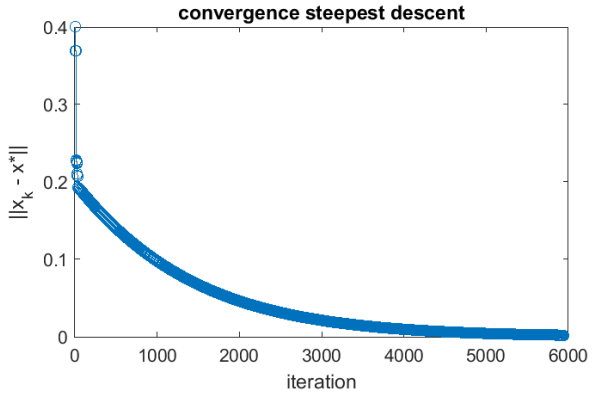


(b) Newton's

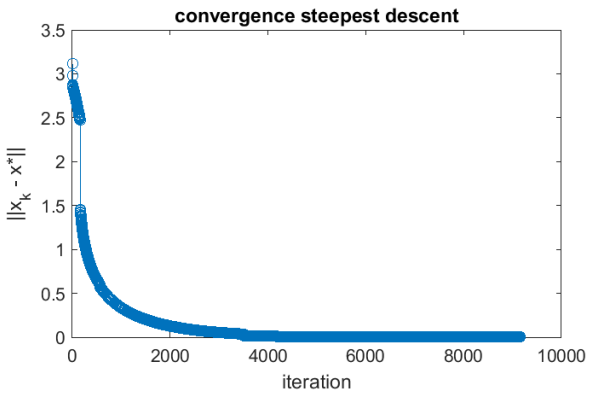
The trajectories of the algorithm's can be explained in a similar fashion to **(b)**. Note that the Newton method is more 'wiggly' since it is following the curvature given by both the first and second derivatives of Rosenbrock function - and hence is using more 'information' about the curvature of the function when tracing out the path.

3.4 (d)

The convergence of the iterates is given by $||x_k - x^*||$ where x^* is the true minimum - i.e. the difference between the iterate value x_k and the true minimum x^*. Plots of the convergence of the steepest descent algorithm from the two different starting points are given below:



(a) $x_0 = (1.2, 1.2)^T$



(b) $x_0 = (-1.2, 1)^T$

Qualitively, we see that the convergence rate for the steepest descent algorithm is extremely slow. From the starting point $x_0 = (1.2, 1.2)^T$ it takes 6000 iterations, and from the starting point $x_0 = (-1.2, 1)^T$ the convergence rate is even slower, taking 9169 iterations to converge. More precisely we note that the rate of convergence for steepest descent appears to be **sublinear**. This is shown by the slope of the graphs and noting that it takes many thousands of iterations for $|x_k - x^*|$ to reduce from 0.1 to 0 from both starting points. However we can also use more quantitative analysis to show why this is true. Note that a sequence $\{x_k\}$ is said to converge sublinearly to x^* if the following condition holds[1]:

$$\lim_{k \to \infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = 1 \tag{1}$$

[1] https://en.wikipedia.org/wiki/Rate_of_convergence

Although we cannot of course take the limit to infinity we found that for the starting point $x_0 = (-1.2, 1.0)^T$ and maximum iterations $k = 9169$, the numerical approximation to equation (1) were: $\frac{|x_{9169} - x^*|}{|x_{9168} - x^*|} = 1$. Therefore we see that condition (1) holds in our approximation of the limit and can therefore conclude that the rate of convergence for steepest descent is sublinear. We also provide the graph of $\frac{|x_{k+1} - x^*|}{|x_k - x^*|^q}$ to show this limit holds more concretely:

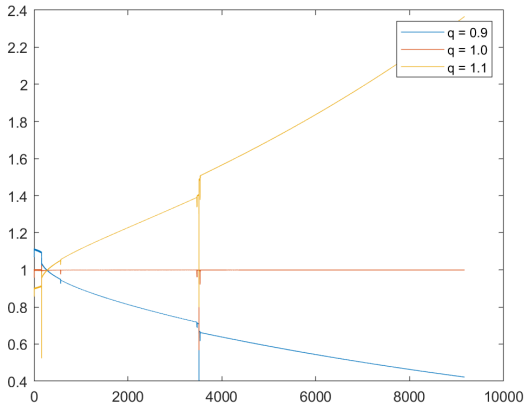
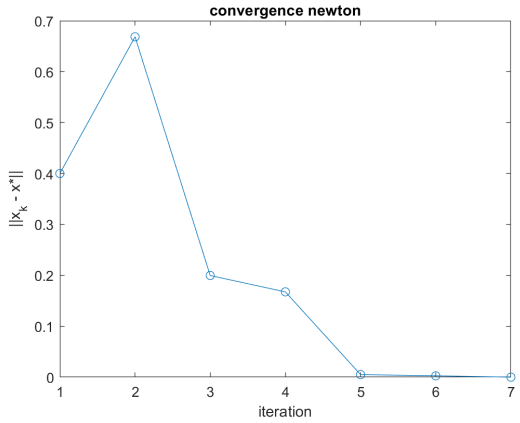


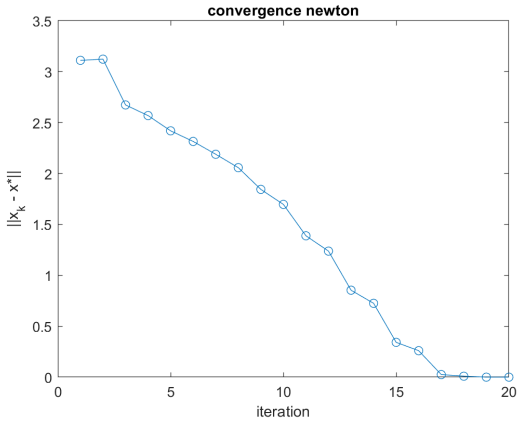
Figure 6: plot of $\frac{|x_{k+1} - x^*|}{|x_k - x^*|^q}$

Note how for $q > 1$ and $q < 1$, the limit $\lim_{k \to \infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^q}$ does not converge to 1, and that $q = 1$ corresponds precisely to sublinear convergence.

Plots of the convergence of Newton's from the different starting points are also shown:



(a) $x_0 = (1.2, 1.2)^T$



(b) $x_0 = (-1.2, 1)^T$

Although it is hard to infer the convergence rate of the iterates (since they are not strictly decreasing), we note that the convergence rate of Newton's appears to be much faster than steepest descent, and the rate seems to be **linear**, since the distance $|x_k - x^*|$ seems to decrease as a linear function with respect to the iterate number k.