COMP0120 Numerical Optimisation

Assignment 1

Date:
FEBRUARY 2022

QUESTION 1

(a)

$$\nabla f(x,y) = \begin{bmatrix} 2\sin x\,(y - \cos x) - 2(y - x) \\ 2(y - \cos x) + 2(y - x) \end{bmatrix}$$

$$\nabla^2 f(x,y) = \begin{bmatrix} 4\sin^2 x + 2y\cos x & 2\sin x - 2 \\ 2\sin x - 2 & 4 \end{bmatrix}$$

(b)

We first note that $f(x,y) \geq 0, \forall x,y$ since we have the sum of two squares. Now,

$$f(x,y) = 0 \iff y = \cos x \text{ and } y = x\,.$$

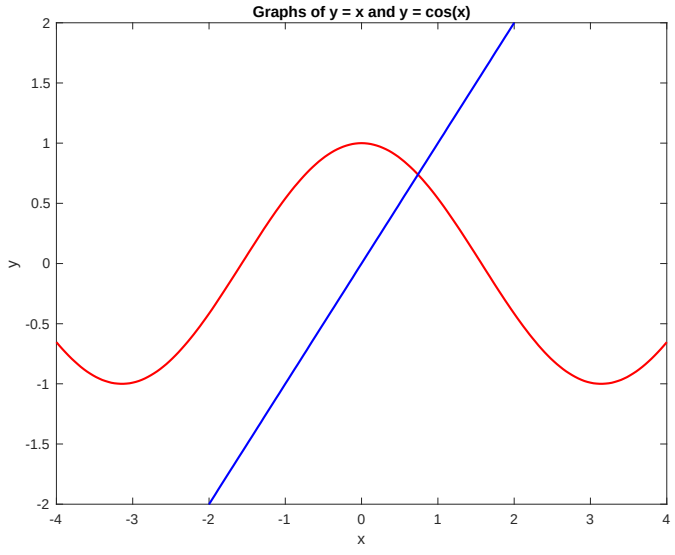
Which we can rewrite as:

$$\cos x - x = 0\,.$$

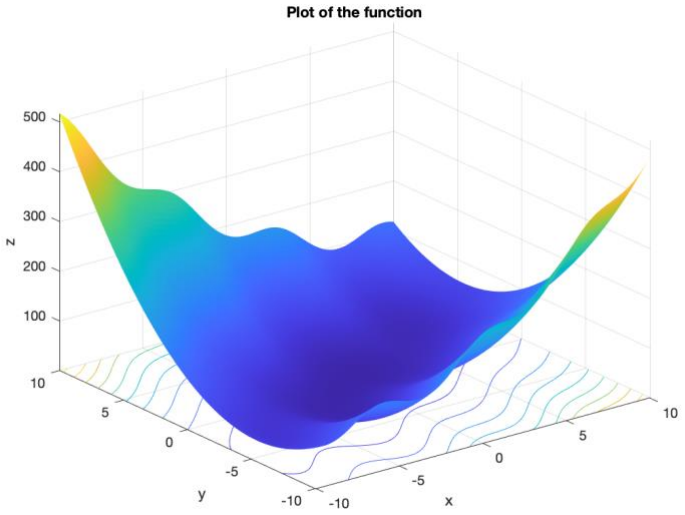
Using the Newton-Raphson method we can find an approximate solution to this equation, and so we have that our minimiser:

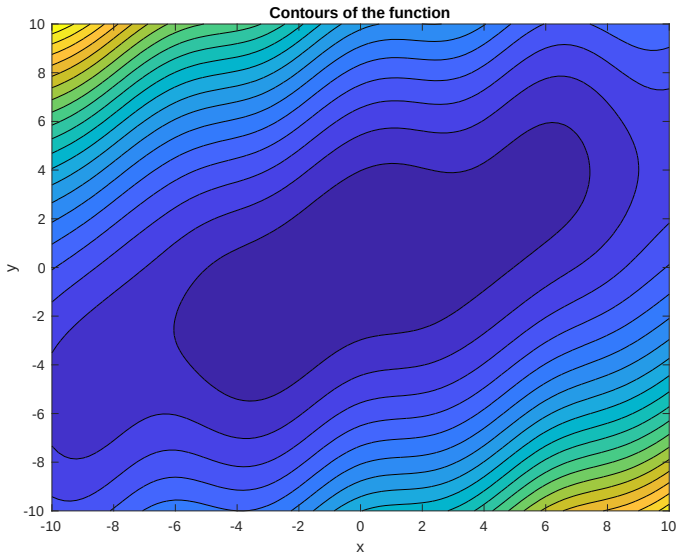
$$x^* \approx [0.739085, 0.739085]^T \quad \text{(6.d.p.)}$$

To test whether our minimiser satisfies the 1st order necessary condition we put x^* back into our computed gradient ∇f. Doing so yields $\nabla f \approx 0$ (within an appropriate tolerance) and hence, x^* satisfies the 1st order necessary condition. To show $x^*$is unique, we plot the lines $y = \cos x$ and $y = x$ to show that they intersect at one unique point.



(c)





To show the level sets are bounded we are given the hint to consider large values of $|x|$ and $|y|$. In doing so, we notice that the term $(y - \cos x)^2 \approx y^2$ since $\cos x$ is bounded between $[-1, 1]$. Hence, our contours simplify to:

$$y^2 + (y - x)^2 = c$$

Upon rearranging, we find that:

$$c(x, y) := (y : x = y \pm \sqrt{c - y^2})$$

We are only interested in the case $(c - y^2) \geq 0$. Hence, we can say that $y \in [-\sqrt{c}, \sqrt{c}]$.

Computing the derivative, we can see that $y \geq 0$ in this region.

We consider now:

$$g_{\pm}(y) := \max_{y \in [-\sqrt{c}, \sqrt{c}]} / \min (y \pm \sqrt{c - y^2})$$

Let us introduce two function definitions:

$$l(y) := g_+(y) = y + \sqrt{c}$$
$$u(y) := g_-(y) = y - \sqrt{c}$$

Using these two definitions we can observe the following inequalities:

$$\min_y l(y) \leq \min_y g_+(y) \leq \min_y g(y)$$
$$\max_y u(y) \geq \max_y g_-(y) \geq \max_y g(y)$$

As a result, we can obtain values for y_{min} and y_{max} , so we can construct our set:

$$\Omega = \{ (x, y) : x \in [l(y_{min}), u(y_{max})], \ y \in [-\sqrt{c}, \sqrt{c}] \}$$

Hence, our level set is bounded.

(d)

We start by taking $\bar{x}, \bar{y} \in \Omega \subset \mathbb{R}^2$ and defining $\bar{y} = \bar{x} + p \implies p = \bar{y} - \bar{x}$. Then taking the integral form of Taylor's theorem, we have:

$$\nabla f(\bar{y}) = \nabla f(\bar{x}) + \int_0^1 \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x}))\,(\bar{y} - \bar{x})\, dt$$

Rearranging and taking norms, we find that:

$$|| \nabla f(\bar{y}) - \nabla f(\bar{x}) || \leq \int_0^1 || \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x})) || \, || (\bar{y} - \bar{x}) || \, dt$$

By introducing $\bar{z} \in \Omega$ we have that:

$$|| \nabla f(\bar{y}) - \nabla f(\bar{x}) || \leq \max_{\bar{z} \in \Omega} \|\nabla^2 f(\bar{z})\| \, || (\bar{y} - \bar{x}) ||, \quad \forall \bar{x}, \bar{y}$$

So far, we have used the L_2 matrix norm, but we are given in the hint that the Frobenius norm is an upper bound to the L_2 matrix norm, so we can say that:

$$|| \nabla f(\bar{y}) - \nabla f(\bar{x}) || \leq \max_{\bar{z} \in \Omega} \|\nabla^2 f(\bar{z})\| \, || (\bar{y} - \bar{x}) || \leq \max_{\bar{z} \in \Omega} || \nabla^2 f(\bar{z}) ||_F \, || (\bar{y} - \bar{x}) ||$$

Hence, we have shown the gradient is Lipschitz continuous.

(e)

Again, let us start by taking $\bar{x}, \bar{y} \in \Omega \subset \mathbb{R}^2$ and defining $\bar{y} = \bar{x} + p \implies p = \bar{y} - \bar{x}$. Then taking the integral form of Taylor's theorem, we have:

$$\nabla f(\bar{y}) = \nabla f(\bar{x}) + \int_0^1 \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x})) \, (\bar{y} - \bar{x}) \, dt$$

Then by applying the gradient and rearranging we have:

$$\nabla^2 f(\bar{y}) - \nabla^2 f(\bar{x}) = \nabla \left(\int_0^1 \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x})) \, (\bar{y} - \bar{x}) \, dt \right)$$

Now we can take the norm on both sides:

$$|| \nabla^2 f(\bar{y}) - \nabla^2 f(\bar{x}) || = || \nabla \left(\int_0^1 \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x})) \, (\bar{y} - \bar{x}) \, dt \right) ||$$

Hence, we have:

$$|| \nabla^2 f(\bar{y}) - \nabla^2 f(\bar{x}) || \leq \int_0^1 || \nabla \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x})) \, dt \, || \, || (\bar{y} - \bar{x}) ||$$

If we multiply and divide the right-hand side by $f(\bar{x})$, we can say that:

$$|| \nabla^2 f(\bar{y}) - \nabla^2 f(\bar{x}) || \leq \int_0^1 || \nabla f(\bar{x}) \, \nabla^2 f(\bar{x} + t(\bar{y} - \bar{x})) \, dt \, || \, || (\bar{y} - \bar{x}) || \, || \frac{1}{f(\bar{x})} ||$$

Which, by part (d), we can see:

$$|| \nabla^2 f(\bar{y}) - \nabla^2 f(\bar{x}) || \leq || \nabla f(\bar{x}) || \max_{\bar{z}} || \nabla^2 f(\bar{z}) || \, || (\bar{y} - \bar{x}) || \, || \frac{1}{f(\bar{x})} ||$$

And now, by Young's inequality:

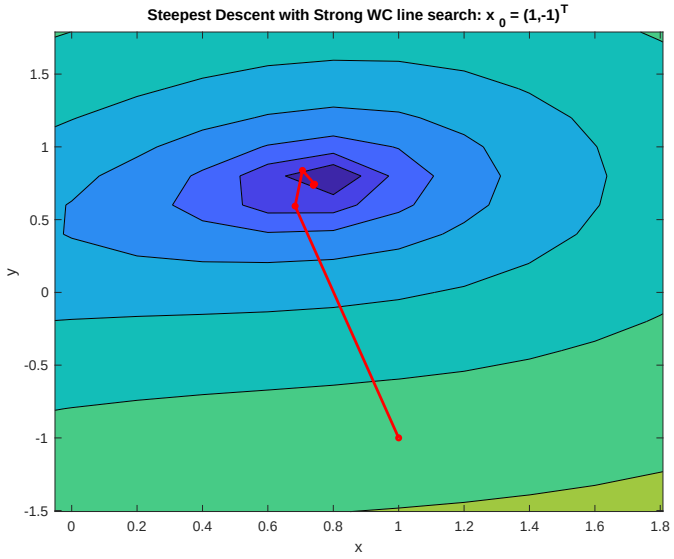
$$|| \nabla^2 f(\bar{y}) - \nabla^2 f(\bar{x}) || \leq || \frac{1}{f(\bar{x})} || \left(\frac{1}{p} || \nabla f(\bar{x}) ||^p + \frac{1}{q} \max_{\bar{z}} || \nabla^2 f(\bar{z}) ||^q \right) || (\bar{y} - \bar{x}) ||$$

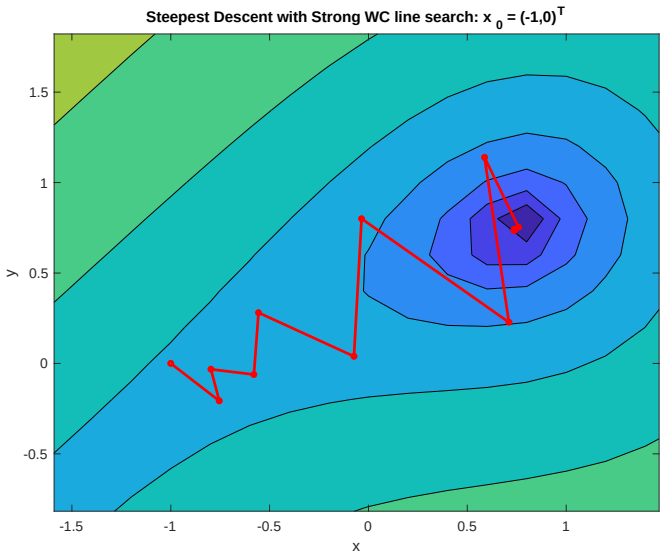
Our minimiser x^* is bounded away from 0, so the first term on the right-hand side is bounded. Similarly, for the terms inside the brackets, we have shown that both $|| \nabla f(\bar{x}) ||$ and $\max_{\bar{z}} || \nabla^2 f(\bar{z}) ||$ are bounded in part (d). Hence the product of these terms together is also bounded. Therefore, we have shown that the Hessian is Lipschitz continuous.

QUESTION 2

(a)

Application of steepest descent to our two initial points: $x_0 = [1,-1]^T$ and $x_0 = [-1,0]^T$.





Where we have used Strong Wolfe Condition line search.

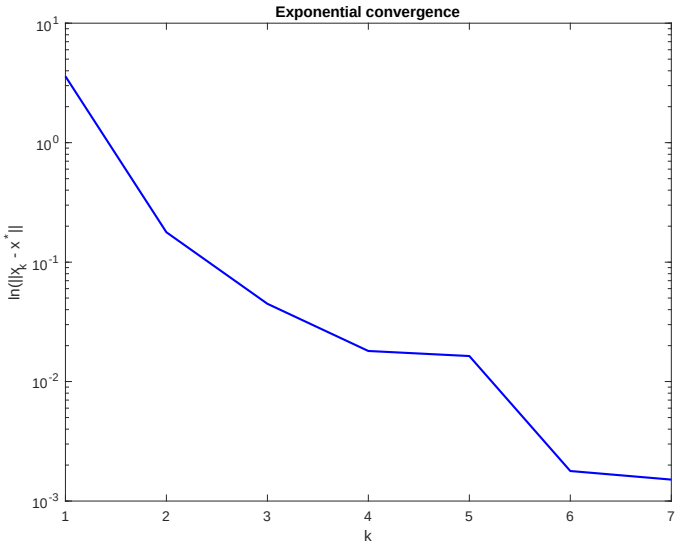
The parameters associated with the line search are:

$$\alpha_0 = 1; \quad c_1 = 10^{-4}; \quad c_2 = 0.9$$

We notice that the point $x_0 = [1,-1]^T$ requires fewer iterations as the steepest descent direction (the direction perpendicular to the contour at each step) is almost directly in line with the global minimum x^*. However, for the point $x_0 = [-1,0]^T$ the contours are more elongated – the point falls within a valley – meaning more iterations are required, and we can clearly see the characteristic zig-zagging.

(b)

A posteriori, we have seen our function converged to the point x*. We examine the convergence rates by considering the error plots.



From the graph we can argue there is a (slightly uneven) linear dependence between $log\, e_k$ *and* k , which indicates exponential convergence and hence implies linear convergence. This is to be expected from the initial point $x_0 = [1, -1]^T$ as the function appears convex (the contours appear more circular than elliptical). This agrees with theoretical predictions since our function $f(x, y)$ is twice continuously differentiable at the minimiser x^* (Question 1 – Part (a)) and we have a symmetric positive definite Hessian with eigenvalues:

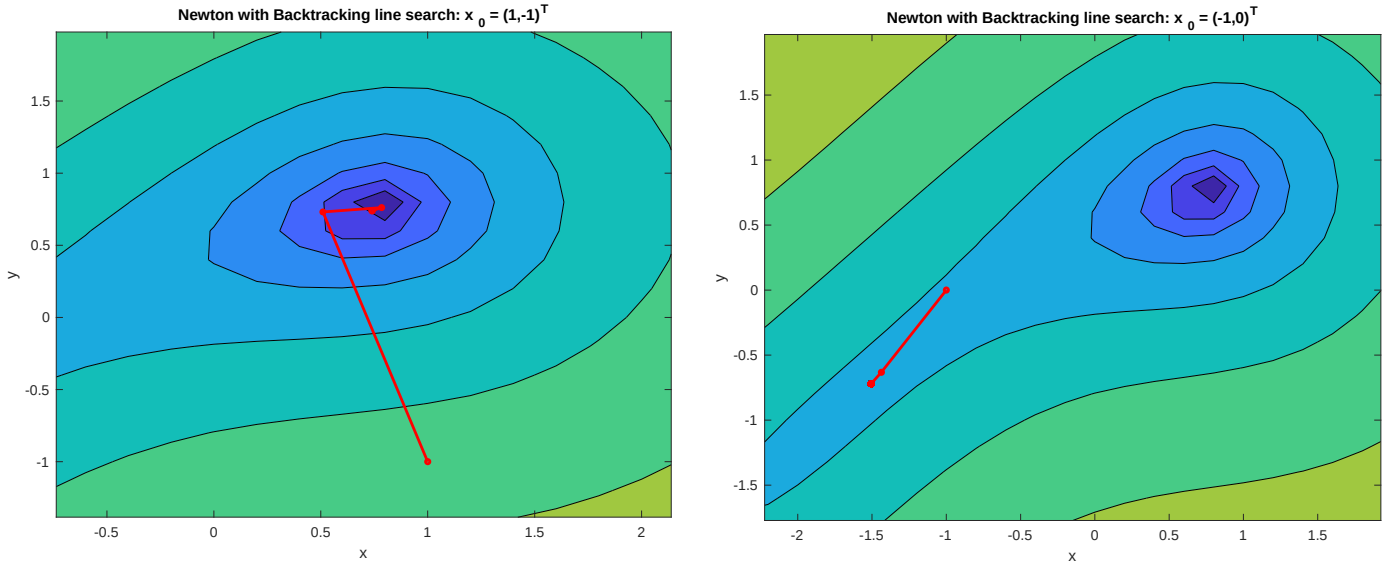
$$\lambda_1 \approx 2.6\,, \qquad \lambda_2 \approx 4.3 > 0\,.$$

As a result, we can estimate the rate of convergence using the formula:

$$\left(\frac{\lambda_2 - \lambda_1}{\lambda_2 + \lambda_1}\right) \approx 0.2464$$

(c)

Application of Newton descent to our two initial points: $x_0 = [1, -1]^T$ and $x_0 = [-1, 0]^T$.



Where we have used Backtracking line search.

The parameters associated with the line search are:

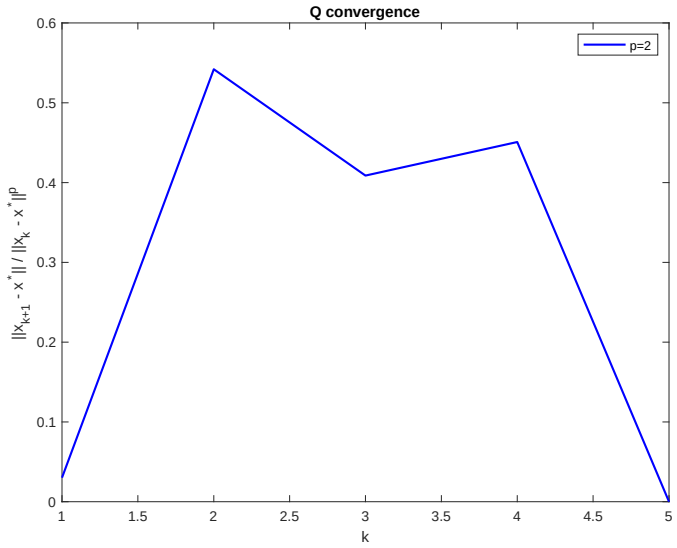
$$\alpha_0 = 1; \quad c_1 = 10^{-4}; \quad \rho = 0.9$$

As we can see from the plots, the two starting points show starkly different behaviours. From the point $x_0 = [1, -1]^T$ we have fast convergence to the minimiser x^*. However, when starting at $x_0 = [-1, 0]^T$ the Newton method takes steps in the wrong direction and fails to converge, regardless of the line search method used.

(d)

We have seen that our Newton direction performs quite differently depending on its starting position, both converging and diverging from our initial points. We discuss the diverging case $x_0 = [-1, 0]^T$ first. According to theory, Newton's method is well behaved and will generate descent directions when the Hessian is positive definite. This may only happen in a specific area around the minimiser x^*, meaning that outside that limit, the Hessian fails to be positive definite, and the method will not converge. Upon closer inspection, our point $[-1, 0]^T$ produces one positive and one negative eigenvalue, leading to the conclusion the Hessian is indefinite and the lack of convergence agrees with the theoretical results.

Our second initial point $x_0 = [1, -1]^T$ does converge a posteriori, and therefore falls within the well-behaved region around x^* (the Hessian is indeed positive definite). We can now examine the error plot.



The graph shows that the Newton method converges quadratically. Once again, this is expected as the point $x_0 = [1, -1]^T$ appears convex, and the same results regarding differentiability and symmetric positive definite Hessian from part (b) hold. In addition, we have shown the Hessian is Lipschitz continuous when sufficiently close the minimiser x^* (Question 1 – Part (e)) and so the theory states that we can expect quadratic convergence using the Newton method in this region.

(e)

The discussion of global convergence relies on the following result by Zoutendijk:

$$\sum_{k \geq 0} cos^2\theta_k \, \|\nabla f(x_k)\|^2 < \infty$$

As a result, it can then be shown that

$$cos^2\theta_k \, \|\nabla f(x_k)\|^2 \to 0$$

This is then used as a component to prove global convergence. Using the Steepest Descent method with Strong Wolfe Conditions line search will satisfy Zoutendijk's result above and hence will globally converge. Newton's method is slightly more complicated and cannot guarantee global convergence without additional requirements. For global convergence of Newton's method, we must have that the actual Hessian B_k is positive definite (as we have seen in our example), and there must be come constant M, such that:

$$\|B_k\| \left\|B_k^{-1}\right\| \leq M$$

This means that the condition number is uniformly bounded, which in turn allows us to guarantee the following strong global convergence:

$$\lim_{k \to \infty} \|\nabla f_k\| = 0$$

Provided those conditions have been satisfied.

QUESTION 3

(a)

Dogleg Implementation MATLAB code.

```
function p = solverCMdogleg(F, x_k, Delta)

% Compute gradient and Hessian
g = F.df(x_k);
B = F.d2f(x_k);

% Compute Newton direction
a = -B\g ;

% Orthonormal basis to check for collinearity
V = orth([g, B\g]);

% Case (1)
% Compute unconstrained solution and check if it lies in the trust region
if a'*a < Delta^2
  % Newton direction lies within trust region so complete full step
  % Compute the solution p
  p = a;
  return;
end
%end

% Case (2)
% If Newton point p_B lies outside the trust region, but collinear to
% gradient
if a'*a >= Delta^2 && size(V,2) == 1
  % Calculate Cauchy point
    gTBg = g'*(B*g);
    if gTBg <= 0
        tau1 = 1;
    else
        tau1 = min(norm(g)^3/(Delta*gTBg), 1);
    end
    p = -tau1*Delta/norm(g)*g;
    return;
end

% Compute p_U
p_U = ((-g'*g)/(g'*B*g))*g ;

% Case (3)
% Newton point p_B lies outside trust region, and p_U lies outside the trust
% region
if norm(p_U) >= Delta
    p = (Delta / norm(p_U)) * p_U ;
    return;
end

% Case (4)
% Newton point p_B lies outside trust region, but p_U lies inside the trust
% region

if norm(p_U) < Delta
    % Find tau that solves
    % || p_U + (tau - 1)(p_B - p_U) ||^2 = Delta_k^2
    tau2 = fzero( @(t) norm(p_U + (t-1)*(a - p_U))^2 - Delta^2 , 1.5);
    % We then put that tau back into
    % p_U + (tau - 1)(p_B - p_U) to find the intersect point and then p =
    % that point
    p = (p_U + (tau2 - 1)*(a - p_U));
    return;
end
```

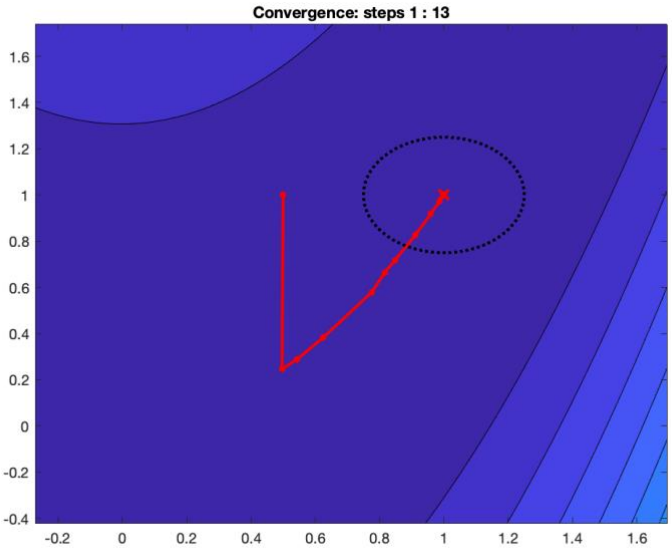
The need to compute the intersection between the trust region and the dogleg only occurs in one specific case. That is, the case where the computed minimum p^U of our approximate function $m_k(p)$ falls within our trust region radius, but the full step p^B lies outside the trust region radius. This means we can find a better step than p^U which lies on the second part of the dog's leg (the line segment from p^U to p^B). We do this by finding the scalar τ that tells us how far to move along the line segment p^U to p^B and then performing a linear

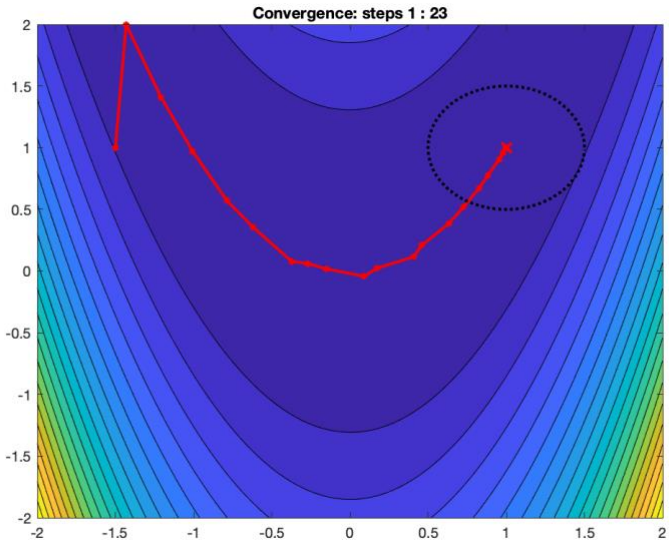
combination of the first part of the dog's leg and an additional component from the second part of the dog's leg, to compute our next step.

(b)

Application of Dogleg Trust Region method to our two initial points:

$$x_0 = [0.5, 1]^T \text{ and } x_0 = [-1.5, 1]^T.$$





Our important parameters for the method are:

$$\eta = 0.2; \quad \Delta = 2; \quad tol = 10^{-6}$$

We have used the gradient stopping condition:

$$\|\nabla f(x_k)\|_\infty < tol\,(1 + |f(x_k)|)$$

The first thing to note is that the method converged in both cases, with the closer point $x_0 = [0.5, 1]^T$ requiring fewer steps. Secondly, we note that in both cases, despite being on the same line, $y = 1$, as the minimiser, the method takes a relatively large step in the perpendicular direction before correcting itself.

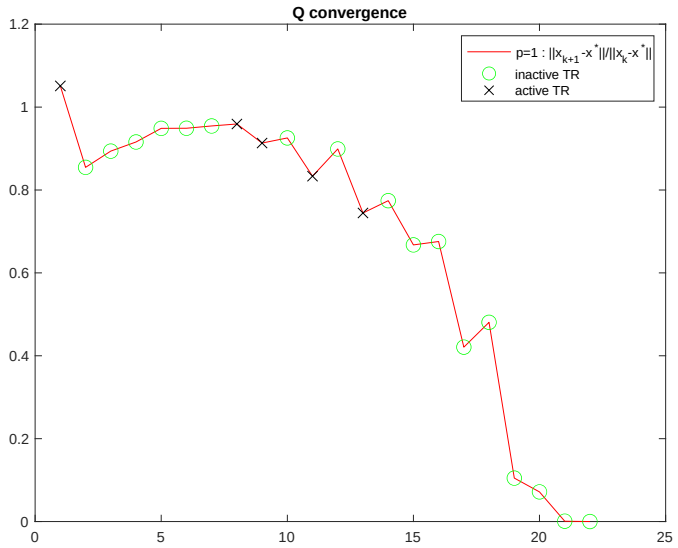
(c)

The error plot, whilst rather jagged, provides some insight into the convergence behaviour of our Dogleg Trust Region method. We can see from step 14 the trust region becomes inactive, and the error tends towards zero for increasing *k*. This implies the convergence rate is superlinear. This agrees with the theoretical results which state that the trust region radius Δ will become inactive, and convergence will be superlinear for all sufficiently large k under certain conditions. Firstly, we must have the following:

- $f(x, y)$ is twice continuously differentiable in a neighbourhood of x^*.
- $\nabla^2 f(x, y)$ is continuous in the neighbourhood of x^*.
- $\nabla f(x^*) = 0$.
- $\nabla^2 f(x^*)$ is positive definite.

All of which can be shown to be true for our function (we proved these results in Tutorial 1).

Secondly, the Dogleg implementation returns at least Cauchy points which satisfy the model reduction criteria. Hence, the conditions are satisfied, and our empirical convergence agrees with the theory.



(d)

To establish global convergence, we must first check whether several conditions hold for our function. Firstly, the function must be bounded below on the level set. Immediately we can see this is the case because our function is composed of two squares. It is also worth noting that our dogleg implementation satisfies the requirement that our steps p_k sufficiently decrease at each step because it will return at least a Cauchy point p^C. In addition, we must be able to bound the norm of our Hessian:

$$\|B_k\| \leq \beta$$

And show our function Lipschitz continuously differentiable on the level set. Again, we have proven both of these statements to be true for the Rosenbrock function (Tutorial 1), and so we can guarantee global convergence of our method. Furthermore, because of our choice of $\eta = 0.2 \neq 0$, we have strong (as opposed to weak) global convergence.

QUESTION 4

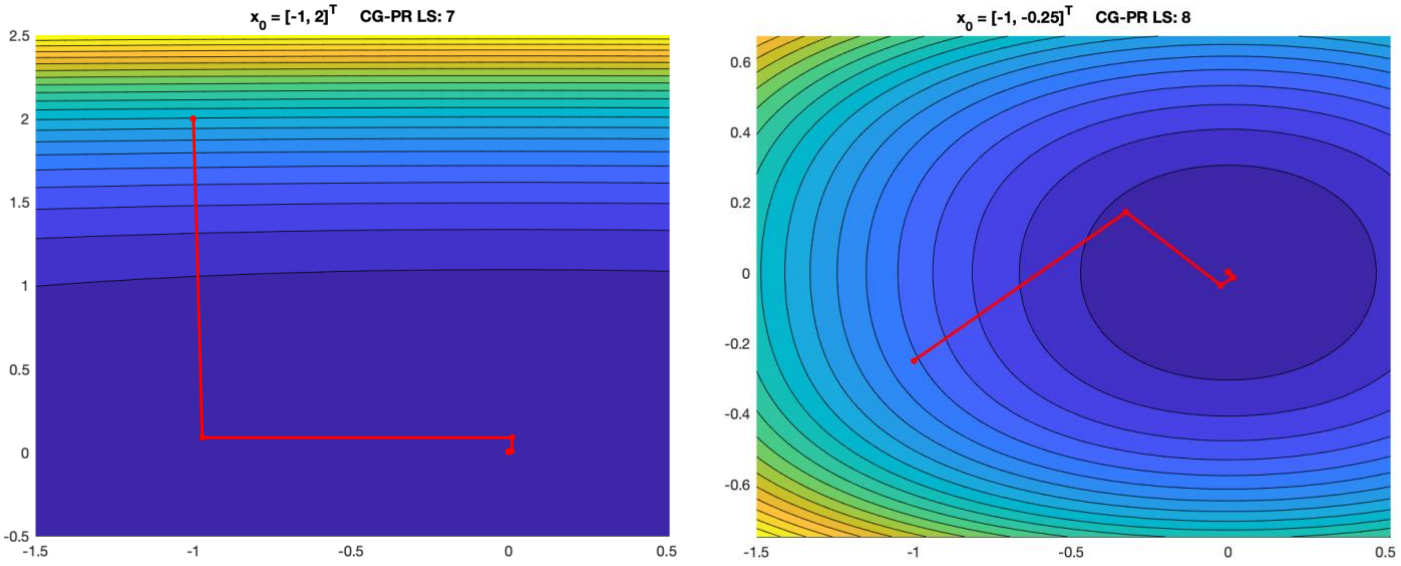
(a)

Polak-Ribière implementation MATLAB code.

```
while (~stopCond && nIter <= maxIter)

    % Call line search given by handle ls to compute step length alpha_k
    alpha(nIter) = lsFun(x, p0, alpha0) ;

    % Update x
    x = x + alpha(nIter)*p0 ;

    % Compute new gradient
    grad(:,nIter+1) = F.df(x);

    % POLAK RIBIERE METHOD
    beta(nIter) = (grad(:,nIter+1)'*(grad(:,nIter+1) - grad(:,nIter))) / norm(grad(:,nIter)).^2 ;

    % Compute new direction p1
    p1 = -grad(:,nIter+1) + beta(nIter)*p0 ;
```

(b)

Application of Polak-Ribière to our two initial points: $x_0 = [-1, 2]^T$ and $x_0 = [-1, -0.25]^T$.



Where we have used the Strong Wolfe Condition line search.

The parameters associated with the line search are:

$$\alpha_0 = 1; \quad c_1 = 10^{-4}; \quad c_2 = 0.1$$

(c)

The primary challenge associated with the Polak-Ribière method is that the steps p_k generated may not necessarily be descent directions. We do not, however, witness this during our implementation. To understand why this is the case, we consider the following equation and examine when p_k will be a descent direction:

$$\nabla f_k^T \, p_k = -\|\nabla f_k\|^2 + \beta_k \nabla f_k^T \, p_{k-1}$$

We need the left-hand side of our equation to be strictly less than zero for p_k to be a descent direction. However, if our β_k is sufficiently negative then the second term on the right-hand side could dominate and

cause ascension. For our function, we do see negative β_k but they are sufficiently small not to cause any issues. Another challenge with the Polak-Ribière method is indefinite cycling. We do observe that use of the Backtracking line search causes the convergence to be erratic and cycle around our minimiser x^* for a period – although it does ultimately converge – but we avoid this by using Strong Wolfe Condition line search instead.

(d)

In its current state we cannot guarantee global convergence because we have used an inexact line search method – the Strong Wolfe Conditions line search. There are ways to adapt the Polak-Ribière method to guarantee global convergence. One approach is to change our definition of β to:

$$\beta_{k+1}^{PR} = max\,(0\,, \beta_k^{PR})$$

We can then adapt the Strong Wolfe conditions to ensure that our direction p_k is always a descent direction. However, there is an alternative approach.

Firstly, we must check whether our function meets the following assumptions:

- The level set of our function $f(x, y)$ is bounded.
- Our function is Lipschitz continuously differentiable in a neighbourhood of the level set.

It is clear from the simplicity of our function that it satisfies these assumptions.

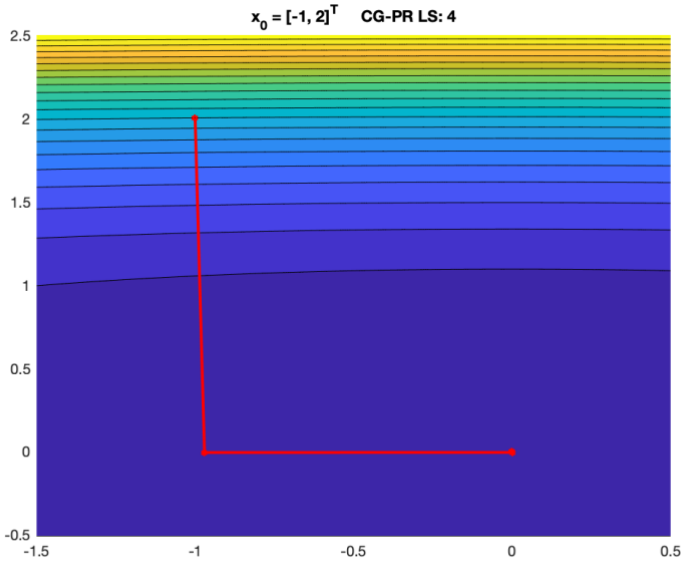
Now we turn our attention to the function itself and notice that it is strongly convex. As a result, by using an exact line search method (instead of our previous inexact line search), we can use Zoutendijk's Theorem to show that:

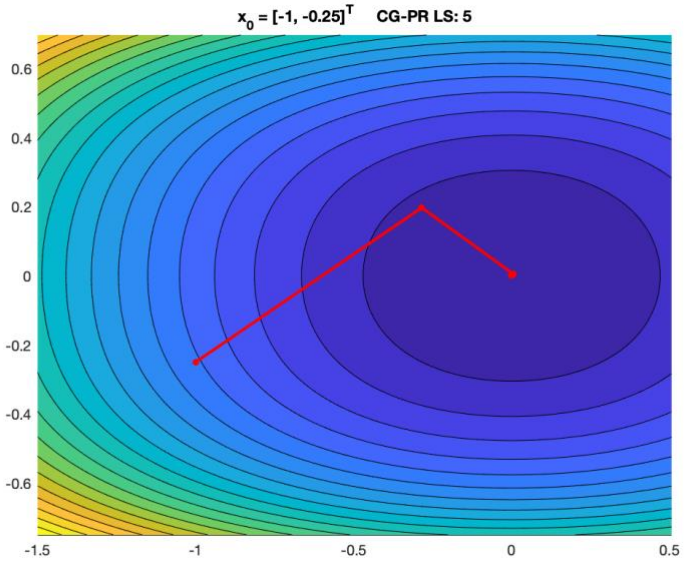
$$\lim_{k \to \infty} \|\nabla f_k\| = 0$$

Which means the Polak-Ribière method will guarantee global convergence if we use an exact line search.

Application of Polak-Ribière, with exact line search, to our two initial points:

$$x_0 = [-1, 2]^T \text{ and } x_0 = [-1, -0.25]^T.$$

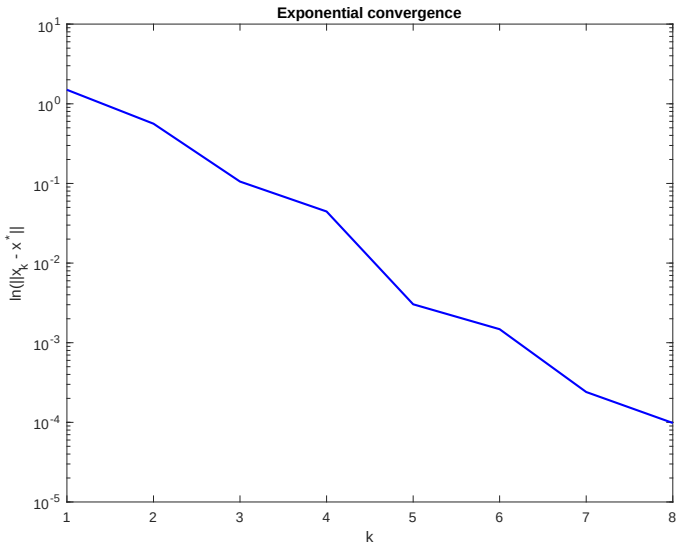




Comparing our result to that in part (b) we can clearly see the method took fewer iterations, in large part due to fewer 'small progress' iterations around the minimiser.

(e)

We have seen that our Polak-Ribière did indeed converge to the minimiser x^* a posteriori. Furthermore, we know that $\nabla f(x^*) = 0$ and the Hessian is positive definite at x^*. We can now examine the convergence rates. In part (d) we have used the exact line search and we should expect linear convergence. However, for part (b) with our inexact line search we have no initial ideas.



As we can see from the graph, we do in fact have linear convergence and this does agree with the theoretical predictions. Whilst linear convergence may not be true for more general cases, because our function is strongly convex the method is well behaved in our region of interest and our Polak-Ribière implementation simply becomes the Fletcher-Reeves method. It is worth noting that improvement on linear convergence (e.g. superlinear) is not possible.

QUESTION 5

(a)

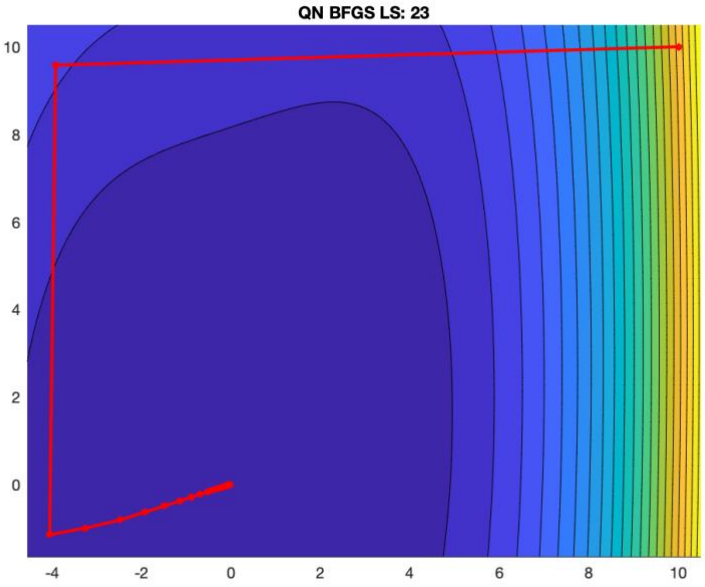
BFGS implementation MATLAB code.

```
% Compute step difference s_k
s_k = x_k - x_k_1 ;

% Compute gradient difference y_k
y_k = g_k - g_k_1 ;

% Compute rho
rho_k = 1 / (y_k'*s_k) ;

% Compute new inverse Hessian H1
H1 = (eye(2) - rho_k*s_k*y_k')*H0*(eye(2) - rho_k*y_k*s_k') + rho_k*(s_k*s_k') ;
```

Our implementation for the updating criteria of H_1 under the BFGS method is particularly efficient as it only performs vector-vector products (apart from our initial guess H_0). This means there are no matrix products or systems, or explicit inverse calculations which require significant computational expense. In addition, our choice of H_0 as the identity matrix means the complexity of each iteration is relatively low at only $O(n^2)$.

(b)

Application of BFGS method to our initial point: $x_0 = [10, 10]^T$.



We have used the Strong Wolfe Condition line search as it serves as a guarantee that our function satisfies the curvature condition:

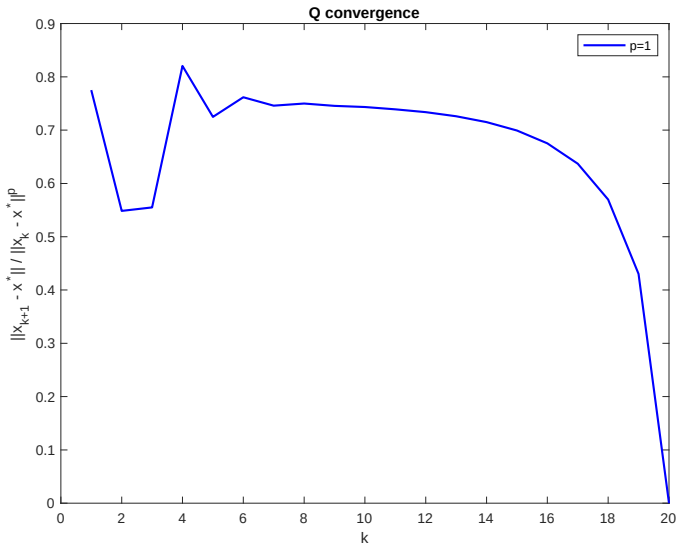
$$y_k^T s_k > 0 .$$

The parameters associated with the line search are:

$$\alpha_0 = 1; \quad c_1 = 10^{-4}; \quad c_2 = 0.9$$

(c)

We have seen that our BFGS method converged to the minimiser x^* a posteriori. We can now examine the associated error plot.



We can see from the graph that the quotient tends towards 0 for increasing k. This implies superlinear convergence. This agrees with the theoretical results that state the BFGS method will achieve superlinear convergence under the following assumptions:

- Our function is twice continuously differentiable.
- The Hessian is Lipschitz continuous at x^*.
- Our sequence x_k converges at a fast enough rate such that:

$$\sum_{k=1}^{\infty} \|x_k - x^*\| < \infty .$$

We can clearly see that our function is twice continuously differentiable. We can assume (and it can be proven) due to the simplicity of our function that the other two conditions hold, and hence our empirical convergence matches the theory.

(d)

Global convergence for the BFGS method is not straightforward. We can guarantee global convergence but only in certain circumstances. For global convergence to occur we require the following assumptions to hold:

- The function $f(x, y)$ must be twice continuously differentiable.
- The level set must convex, which means we can find $m, M > 0$ such that $\forall\, q \in \mathbb{R}^n$:

$$m\|q\|^2 \leq q^T \nabla^2 f(x)\, q \leq M\|q\|^2$$

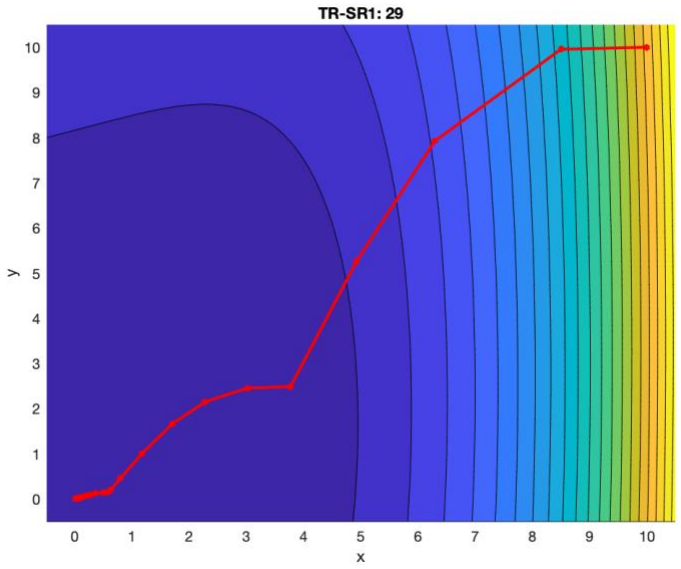
If our initial point x_0 satisfies these assumptions, and we choose our initial Hessian approximation B_0 to be symmetric positive definite, then using Zoutendijk's result it can be shown:

$$\lim_{k\to\infty} inf \left\|\nabla f(x_k)\right\| \to 0$$

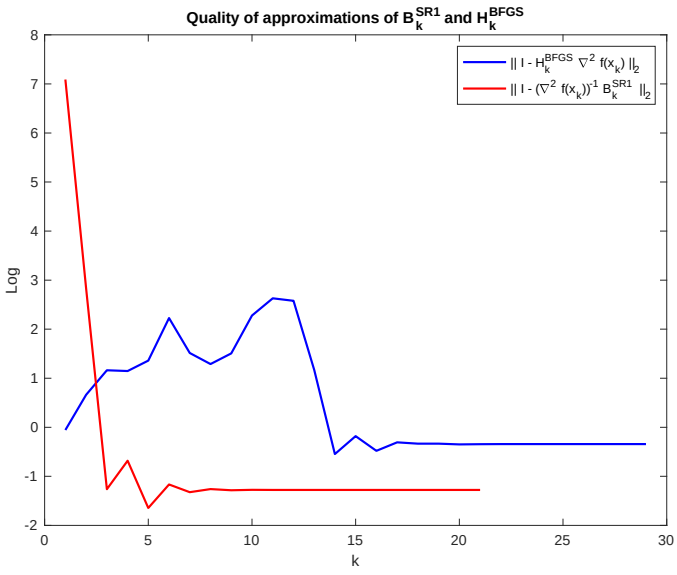
As we can only show the infimum of the gradient norms tends to zero, this is equivalent to weak global convergence.

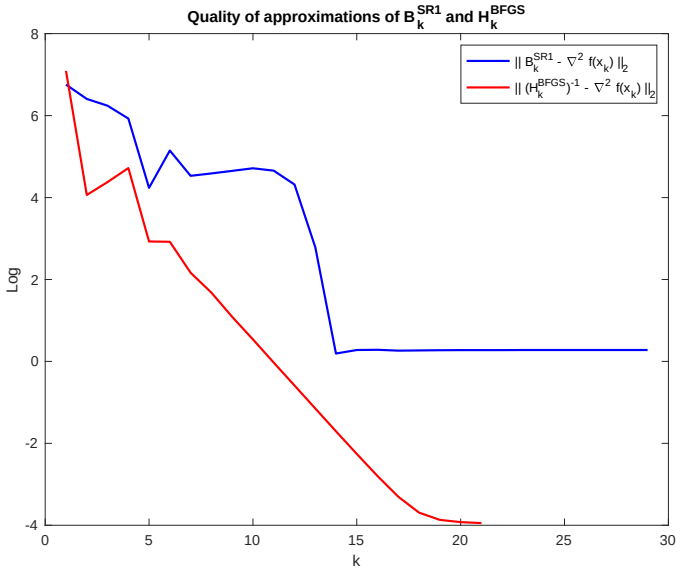
(e)

Application of SR-1 method to our initial point: $x_0 = [10, 10]^T$.



We can see from the graph that our SR-1 application does indeed converge to the minimiser x^*, however, the path towards convergence differs from our BFGS approach. Of course, this is expected as it is a trust region method. We now turn our attention to the quality of the approximate Hessians B_k and inverses H_k.





As we can see from the graphs, both methods give good approximations to the Hessian and its inverse, however, the BFGS method provides a superior estimate. This is because we used the Strong Wolfe Conditions line search during our BFGS implementation and as a result, we capture some degree of curvature at each update. From theory this means the BFGS method shows good self-correcting properties and therefore we have accurate approximations of the inverse Hessian H_k.

The SR-1 method also provides a good approximation, however, it seems to plateau for larger values of k. To explore this further, we can examine the condition numbers for both the approximate Hessian and the inverse. Our computed condition number for H_k terminates around the order 10^3, while B_k terminates around the 10^5. This could be cause for alarm, and we could mistakenly conclude that our approximations are poor, however, upon closer inspection we notice that the true Hessian at our true minimiser is singular:

$$\nabla^2 f(x^*) = \begin{pmatrix} 2 & -6 \\ -6 & 18 \end{pmatrix}$$

Hence, our condition number will increase exponentially as our approximation approach the true value.