NUMERICAL OPTIMISATION COURSEWORK I

ID: 21118015

Exercise 1

We are given a function $f:\mathbb{R}^2 \to \mathbb{R}$

$$f(x,y)=(y-\cos x)^2+(y-x)^2$$

(a) From the expression of f, we can easily get the gradient ∇f and the Hessian $\nabla^2 f$ as follows:

$$\nabla f=[f_x,f_y]^{\mathrm{T}}=[2y(\sin x-1)+2x-\sin 2x,\ 4y-2\cos x-2x]^{\mathrm{T}}$$

$$\nabla^2 f=\begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}=\begin{bmatrix} 2y\cos x+2-2\cos 2x & 2(\sin x-1) \\ 2(\sin x-1) & 4 \end{bmatrix}$$

(b) We use the steepest descent method to find the minimizer x^*. We initialize the start point x_0=(1,-1), setting the negative gradient direction of current point $-\nabla f(x_k)$ as the decent direction p_k. Hence, we can build the iteration formula:

$$x_{k+1}=x_k+\alpha_k p_k\ ,\ k\geqslant 0$$

where α_k is the step length of every time iteration, we make sure that α_k follows the strong Wolfe condition with c_1=10^{-4}, c_2=0.1. The stopping criteria is satisfied after 6 times iteration, then we find the optimal point x_6=(0.7390,0.7390). Then we calculate the gradient value of x_6 and obtain

$$\nabla f(x_6)=[-1.9193\times 10^{-4},-2.8495\times 10^{-4}]^{\mathrm{T}}$$

Since $f:\mathbb{R}^2\to\mathbb{R}$ is continuously differentiable, and

$$\|\nabla f(x_6)\|_2=3.4356\times 10^{-4}<\varepsilon,\ \varepsilon=1\times 10^{-3}$$

we can conclude that $\nabla f(x_6)\approx[0,0]^{\mathrm{T}}$, which means that the numerical minimizer satisfies the 1st order necessary condition and x^*=x_6. Now we demonstrate that the x^* is unique. It is obvious that $f\geqslant 0$ and $f=0$ iff $\begin{cases} y-\cos x=0 \\ y-x=0 \end{cases}$ (*)

The equation system (*) has the unique solution x^*=(0.7391,0.7391), so we conclude that f takes on a minimal value 0 at x^* and the minimizer x^* is unique.

(c) We can easily observe that the function is strongly convex and level set $L=\{\boldsymbol{x}\in\mathbb{R}^2: f(\boldsymbol{x})\leqslant f(\boldsymbol{x}_0)\}$ is bounded in the limit $\|\boldsymbol{x}\|$ is large from the plot below.

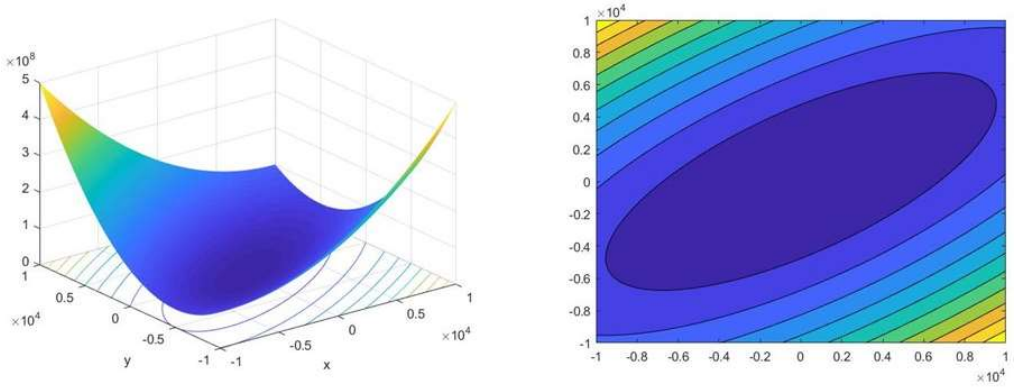


Figure 1. Objective function(left) and its' contour(right)

(d) It is obvious that f is twice continuously differentiable, so we have integral form of Taylor's theorem

$$\int_0^1 \nabla^2 f(x+tp)\,p\,dt$$

Then we assume that

$$\boldsymbol{h}(t)=\nabla^2 f(x+tp)\,p=[h_1(t),\ h_2(t)]^{\mathrm{T}},\ \boldsymbol{h}(t)\in\mathbb{R}^2$$

We know that h_1(t), h_2(t) are continuous in [0,1], so $\exists\ \bar{t}\in[0,1]$ h_i(t) submits to

$$\int_0^1 h_i(t)\,dt=h_i(\bar{t})\,(1-0),\ i=1,2$$

Hence,

$$\begin{aligned}\|\nabla f(x+p)-\nabla f(x)\|_2&=\left\|\int_0^1\nabla^2 f(x+tp)\,p\,dt\right\|_2=\left\|\nabla^2 f\left(x+\bar{t}p\right)p\right\|_2\\&\leqslant\left\|\nabla^2 f\left(x+\bar{t}p\right)\right\|_2\|p\|_2\\&\leqslant\left\|\nabla^2 f\left(x+\bar{t}p\right)\right\|_F\|p\|_2\\&:=L\|p\|_2\end{aligned}$$

where $L=\left\|\nabla^2 f\left(x+\bar{t}p\right)\right\|_F$, thus we can demonstrate that the gradient of f is locally Lipschitz continuous.

(e) In order to show that the Hessian $\nabla^2 f$ is locally Lipchitz continuous, we need to make sure

$$\forall\ N>0,\ \ \exists K_N>0,\ \ \forall\boldsymbol{x}_1=[x_1,y_1]^T,\boldsymbol{x}_2=[x_2,y_2]^T\in\boldsymbol{B}(0,N),\ \text{it holds that}$$
$$\|\nabla^2 f(\boldsymbol{x}_1)-\nabla^2 f(\boldsymbol{x}_2)\|\leqslant K_N\|\boldsymbol{x}_1-\boldsymbol{x}_2\|$$

Proof:

In this case, due to property of matrix norm,

$$\begin{aligned}\|\nabla^2 f(\boldsymbol{x}_1)-\nabla^2 f(\boldsymbol{x}_2)\|_2 &\le \|\nabla^2 f(\boldsymbol{x}_1)-\nabla^2 f(\boldsymbol{x}_2)\|_F \\ &= 2\sqrt{(|\cos 2x_2-\cos 2x_1+y_1\cos x_1-y_2\cos x_2|)^2+2(|\sin x_1-\sin x_2|)^2} \\ &= 2\sqrt{I^2+2II^2}\end{aligned}$$

Applying *Cauchy inequality and Triangular inequality* for *I*

$$\begin{aligned}&|\cos 2x_2-\cos 2x_1+y_1\cos x_1-y_2\cos x_2| \\ &=|(\cos 2x_2-\cos 2x_1)+y_1(\cos x_1-\cos x_2)+(y_1-y_2)\cos x_2| \\ &\le |\cos 2x_2-\cos 2x_1|+|y_1|\,|\cos x_1-\cos x_2|+|y_1-y_2|\,|\cos x_2|\end{aligned}$$

We can deduce that

$$|\cos x_2-\cos x_1|=\left|-2\sin\left(\frac{x_2+x_1}{2}\right)\sin\left(\frac{x_2-x_1}{2}\right)\right|\le 2\left|\sin\left(\frac{x_2+x_1}{2}\right)\right|\left|\sin\left(\frac{x_2-x_1}{2}\right)\right|\le |x_2-x_1|$$

$$|\sin x_2-\sin x_1|=\left|-2\cos\left(\frac{x_2+x_1}{2}\right)\sin\left(\frac{x_2-x_1}{2}\right)\right|\le 2\left|\cos\left(\frac{x_2+x_1}{2}\right)\right|\left|\sin\left(\frac{x_2-x_1}{2}\right)\right|\le |x_2-x_1|$$

Therefore $I\le 2|x_2-x_1|+(N+1)\,|y_2-y_1|,\ II\le |x_2-x_1|,$

$$\begin{aligned}\|\nabla^2 f(\boldsymbol{x}_1)-\nabla^2 f(\boldsymbol{x}_2)\|_2 &\le 2\sqrt{(2|x_2-x_1|+(N+1)\,|y_2-y_1|)^2+2(|x_2-x_1|)^2} \\ &= 2\sqrt{6(|x_2-x_1|)^2+4(N+1)\,|x_2-x_1|\,|y_2-y_1|+(N+1)^2(|y_2-y_1|)^2}\end{aligned}$$

Due to *young inequality(p=2)*, $|x_2-x_1|\,|y_2-y_1|\le \frac{(|x_2-x_1|)^2}{2}+\frac{(|y_2-y_1|)^2}{2},$

Hence,

$$\begin{aligned}\|\nabla^2 f(\boldsymbol{x}_1)-\nabla^2 f(\boldsymbol{x}_2)\|_2 &\le 2\sqrt{6(|x_2-x_1|)^2+2(N+1)\left((|x_2-x_1|)^2+(|y_2-y_1|)^2\right)+(N+1)^2(|y_2-y_1|)^2} \\ &= 2\sqrt{(2N+8)\,(|x_2-x_1|)^2+(N^2+4N+3)\,(|y_2-y_1|)^2} \\ &\le 2\sqrt{N^2+6N+11}\sqrt{(|x_2-x_1|)^2+(|y_2-y_1|)^2} \\ &:= K_N\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2\end{aligned}$$

That means there exists $K_N=2\sqrt{N^2+6N+11}$ to guarantee the Hessian is locally Lipschitz continuous.

Exercise 2

(a) We utilize steepest descent method to find the optimal minimizer. Strong Wolfe condition is considered to calculate the step length. In this exercise, we set initial step length α_0=1 and c_1=10^{-4}, c_2=0.1 in line search to make sure that step length α satisfies

$$\begin{cases} f(x_k+\alpha p)\le f(x_k)+c_1\alpha p^{\mathrm{T}}\nabla f(x_k) \\ |p^{\mathrm{T}}\nabla f(x_k+\alpha p)|\ge c_2|p^{\mathrm{T}}\nabla f(x_k)| \end{cases}$$

Also, we set $\varepsilon=10^{-3}$ and apply it into our stopping criterion $\frac{\|x_{k+1}-x_k\|}{\|x_k\|} \leq \varepsilon$,

which means iteration stops when this criterion is satisfied.

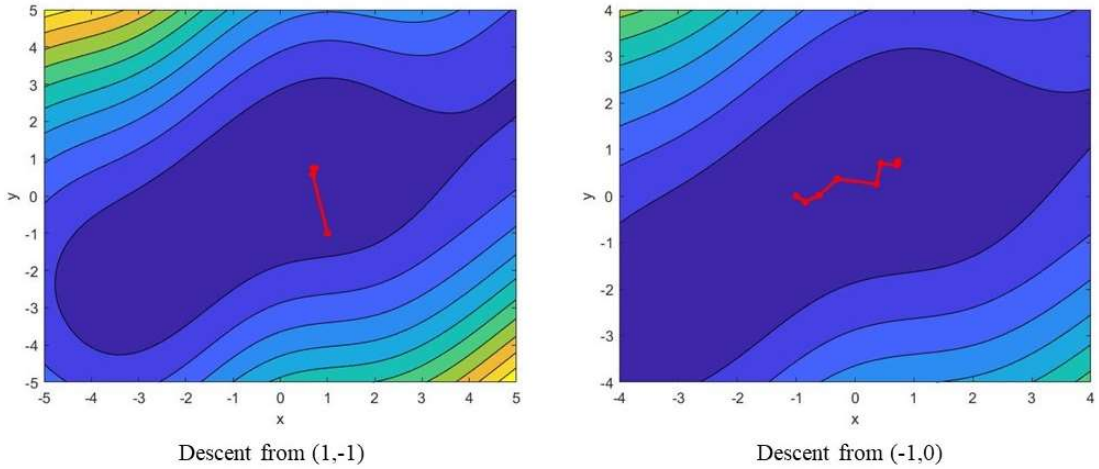


Fig 1. Iteration pot based on steepest descent method

We can observe that more iterations are experienced when our method starts from (-1,0), i.e iterating from (-1,0) converges less efficiently than (1,-1). We can also easily obverse from the plot that the iteration trajectory from (-1,0) shows a zig-zag characteristic, which affects the convergence efficiency.

(b) We can plot relevant error plot based on the posterior: optimal minimizer of the function is (0.7391,0.7391) both for case staring from (1,-1) or (-1,0).

Starting from (1,-1):

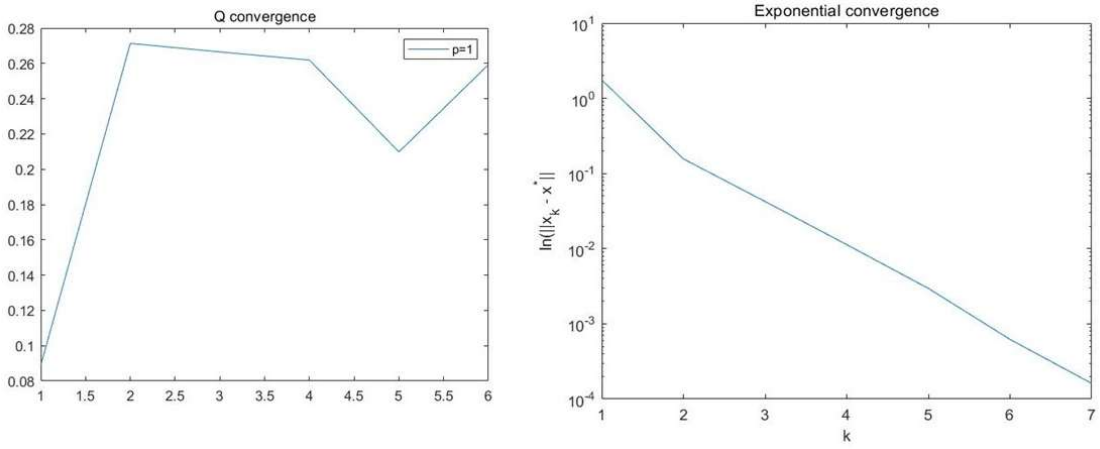


Fig 2. Convergence plot (starting point [1,-1])

From Q convergence plot, we find that $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \leqslant 0.28$ and from algebraic convergence plot, a linear trend $\ln\|x_k-x^*\| = \ln c + k\ln(0.2263)$ is shown, implying that the method guarantees linear convergence with 0.2263 convergence rate. Theoretically, steepest descent method possesses linear convergence with $\frac{\lambda_n-\lambda_1}{\lambda_n+\lambda_1}=0.2465$, where $0<\lambda_1\leqslant\lambda_2\leqslant\ldots\leqslant\lambda_n$ are the eigenvalue of $\nabla^2 f(x^*)$. Hence, our empirical convergence rate is basically corresponding with theoretical result.

Starting from (-1,0):

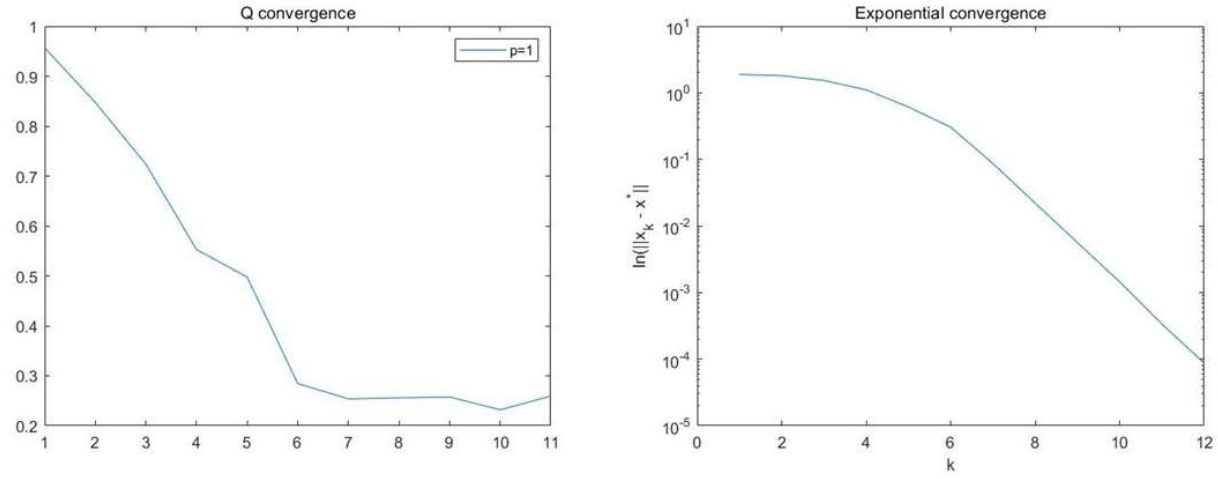


Fig 3. Convergence plot (starting point [-1,0])

When the iteration starts from (-1,0), the exponential convergence plot still approximates to a straight line, with slope equaling to ln(0.2495) when k is sufficient large. So, the linear convergence with rate 0.2495 is applicable for method initialing from (-1,0) empirically, which is also very close to the theoretical prediction. Essentially, the objective function in this case is a twice continuously differentiable nonlinear function and $f(x^*)$ satisfied sufficient conditions, so the linear convergence with theoretical rate 0.2465 is guaranteed.

(c) We still use the strong Wolfe Condition as a line search strategy for Newton method. We set initial step length α_0=10/9, c_1=10^{-4}, c_2=0.1, and set the tolerance for stopping iteration as ε=10^{-3}.

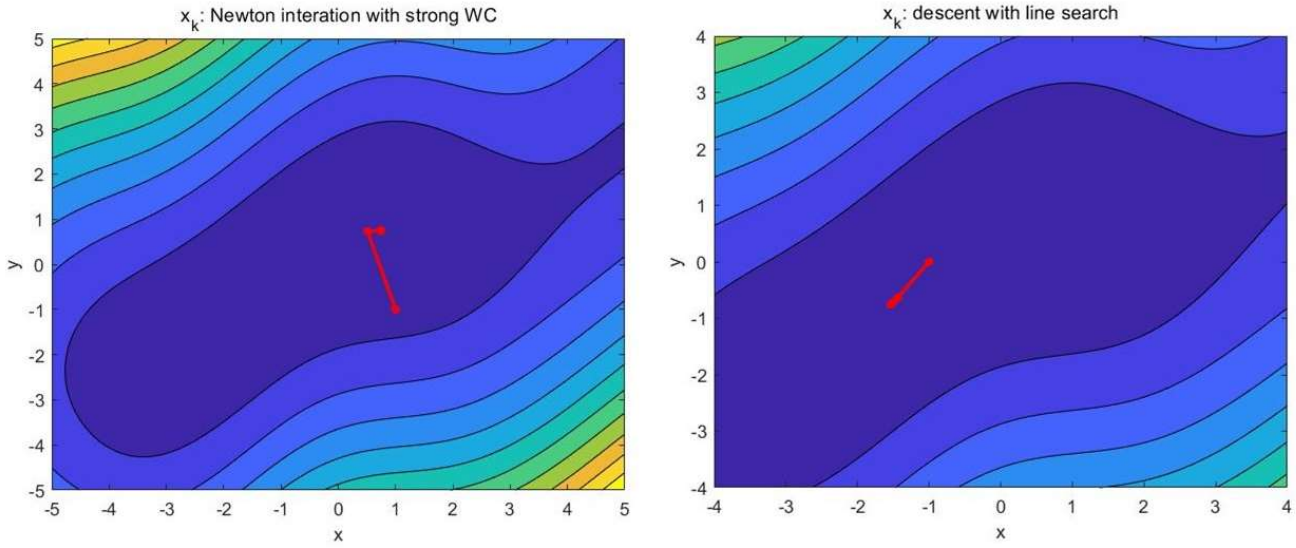


Fig 4. Iteration plot based on Newton method

From the iteration plot, we find that the characteristic zig-zag happened in steepest descent iteration plot has been eliminated. For starting point (1, -1), the convergence is very efficient, it only takes four steps to find the optimal minimizer. However, when we set starting point as (-1,0), which is near the saddle point (-1.4886, -0.6983), the iteration converges to the saddle point rather than minimizer finally, which means that Newton direction is not the descent direction in this case. So, it is obvious that the convergence of Newton method depends highly on the starting point selection.

(d) Figure 5 shows that $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^2} \leqslant M$ where $M=0.45>0$, indicating Newton method with starting point (1,-1) guarantees a quadratic convergence to x^*=(0.7391,0.7391). At the same time, the Newton method iteration stops because it is up to the maximum iteration number limitation. On the other word, convergence

to x^* is not guaranteed under this case.

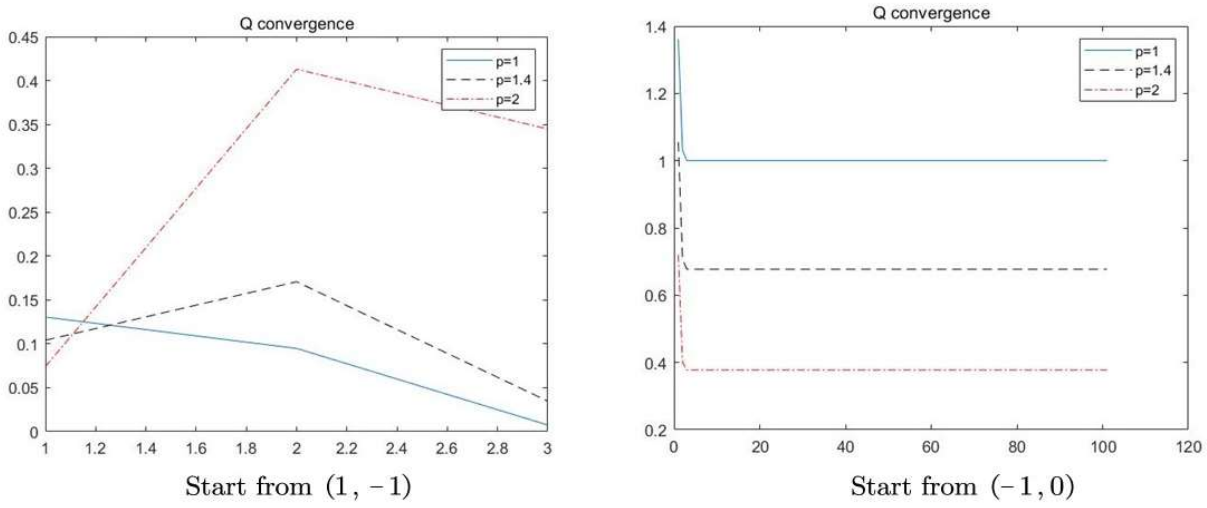


Fig 5. Convergence plot based on each starting point

In theory, we know that f is twice continuously differentiable. In the neighborhood of x^* satisfying the sufficient conditions, f is Lipschitz continuous , $\nabla^2 f$ is positive definite and α_0=10/9 makes sure the step length α_k=1 is admissible for all $k \geqslant k_0$. All the conditions above are satisfied when Newton method starts from (1, -1), so quadratic convergence holds like empirical prediction. For (-1,0) case, the Hessian matrix at saddle point (-1.4886, -0.6983) is not S.P.D, so that the Newton direction becomes not a descent direction, causing the model not converging to x^*.

(e) **Steepest descent method:**

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k),\ k = 0, 1 \ldots$$

(1) $p_k = -\nabla f(x_k)$ is a descent direction and α_k observes strong Wolfe conditions.

(2) f is bounded below on $\mathbb{R}^2$ and is continuously differentiable in an open set M containing the level set. (Due to ex1(d))

(3) ∇f is locally Lipschitz continuous on M. (Due to ex1(e))

Following Zoutendijk lemma, the model is global convergence.

Newton method based on line search:

$$x_{k+1} = x_k - \alpha_k (\nabla^2 f(x_k))^{-1} \nabla f(x_k),\ k = 0, 1 \ldots$$

The convergence of Newton method highly depends on the property of Hessian matrix $\nabla^2 f(x_k)$. If $\nabla^2 f(x_k)$ is not S.P.D on some iteration point, the iteration direction will not be a descent direction. Hence, the global convergence is not guaranteed based on the method.

Exercise 3

(a) The implementation for dogleg trust region method is as follows:

```
01. function p = solverCM2dDogleg(F, x_k, Delta)
02. % Compute gradient and Hessian
03. g = F.df(x_k);
04. B = F.d2f(x_k);
05. gTBg = g'*(B*g);
06. pU=(-(g'*g)/gTBg)*g;
07. pB=-B\g;
08. npB=sqrt(pB'*pB);
09. npU=sqrt(pU'*pU);
10. % Orthonormalize the 2D projection subspace
11. V = orth([g, B\g]);
12. % Check if gradient and Newton steps are collinear. If so return Cauchy point.
13. if size(V,2) == 1
14.   % Calculate Cauchy point
15.   if gTBg <= 0
16.     tau = 1;
17.   else
18.     tau = min(norm(g)^3/(Delta*gTBg), 1);
19.   end
20.   p = -tau*Delta/norm(g)*g;
21. else
22.     % Discuss different cases
23.     if npB<=Delta
24.         t=2;
25.     elseif npU>=Delta
26.         t=Delta/npU;
27.     else
28.         pB_U=pB-pU;
29.         % Solve for the intersection point between trust region and the dogleg path
30.         t=(-pU'*pB_U+sqrt((pU'*pB_U)^2-pB_U'*pB_U*(pU'*pU-Delta^2)))/(pB_U'*pB_U);
31.         t=t+1;
32.     end
33.     % Descent direction
34.     if t>=0 && t<=1
35.         p=t*pU;
36.     elseif t>=1 && t<=2
37.         p=pU+(t-1)*(pB-pU);
38.     end
39. end
40. end
```

Line 29-31 shows how to solve for the intersection point between trust region and dogleg path. We let

$$\begin{cases} \tilde{p}(\tau) = p^U + (\tau - 1)\left(p^B - p^U\right) \\ \left\| \tilde{p}(\tau) \right\|_2 = \Delta^2 \end{cases}, \quad 1 \leqslant \tau \leqslant 2$$

Therefore, we obtain

$$\left(p_B - p_U\right)^T \left(p_B - p_U\right)(\tau - 1)^2 + 2\left(p_B - p_U\right)^T p_U (\tau - 1) + {p_U}^T p_U - \Delta^2 = 0$$

Solving this quadratic equation, we get

$$\tau_0 = \frac{-\left(p_B - p_U\right)^T p_U + \sqrt{\left(\left(p_B - p_U\right)^T p_U\right)^2 - \left(p_B - p_U\right)^T p_U {p_U}^T p_U}}{\left(p_B - p_U\right)^T \left(p_B - p_U\right)} + 1$$

Therefore, $\tilde{p}(\tau_0)$ is the intersection point.

(b) We set step acceptance relative progress threshold η=0.2, trust region radius Δ=3 and stopping tolerance as 10^{-6} for dogleg trust region method and apply it to the Rosenbrock function with two starting points (0.5,1)

and (-1.5,1).

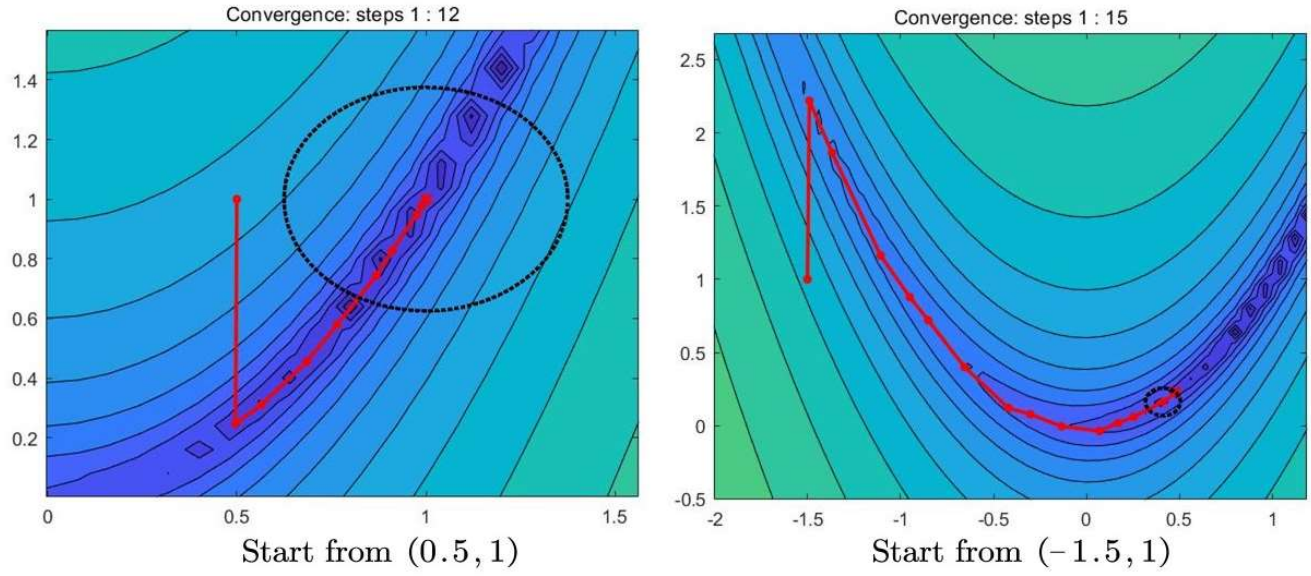


Fig 6. Iteration plot based on Dogleg trust region method

We can observe clearly that for both starting point, the size of trust region changes automatically based on whether the step length is acceptable. The iteration jumps over the local minimizer and converges to the global minimizer, without zig-zag characteristic.

(c)

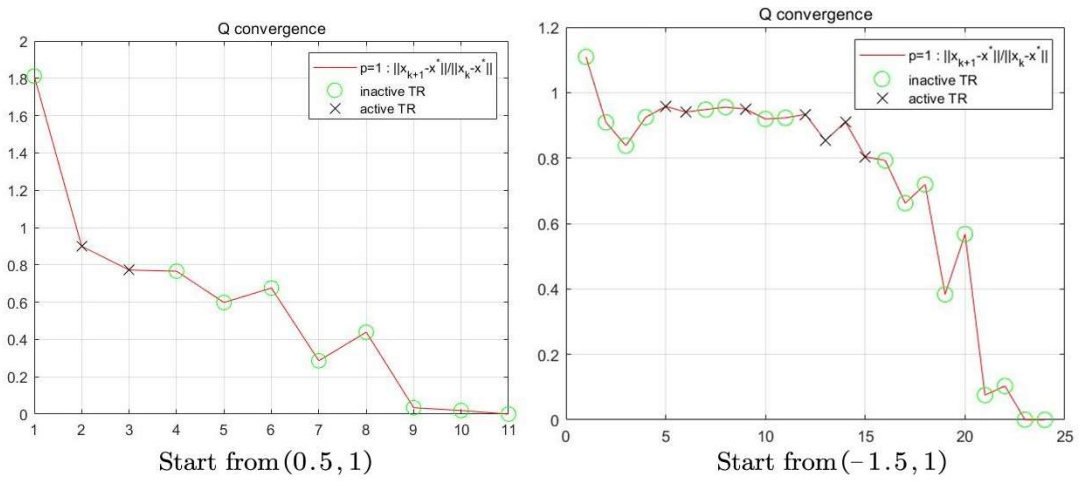


Fig 7. Convergence plot based on each starting point

From the plot above, we predict that the method is superliner convergence, due to $\lim_{k\to\infty} \frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|}=0$.

Theoretically, Rosenbrock function's strictly convex property ensures f is twice Lipschitz continuously differentiable in the neighborhood of a point x^* where the second order sufficient conditions are satisfied. Also, the method chooses steps p_k which satisfies the Cauchy point with sufficient reduction. Under the circumstance, the trust region bound Δ_k becomes inactive when $k \to \infty$ and the superliner convergence is ensured as shown in plot.

(d) **Dogleg trust region method with Cauchy point:**

(1) The objective Rosenbrock function f is strictly convex, so $\forall\ x_k \in \mathbb{R}^2,\ \|\nabla^2 f(x_k)\| \leqslant L,\ L \in \mathbb{R}_+$.

(2) f is bounded below on the level set $S=\{x:\ f(x)\leqslant f(x_0)\}$ and Lipschitz continuously

differentiable in $N(S,R_0),\ R_0>0$

(3) All the approximate solutions p_k of CM satisfies sufficient reduction and $\|p_k\| \leqslant \Delta_k$

Hence, $\lim_{k\to\infty} g_k = 0$ as $\eta \in (0,0.25)$. In this exercise, η=0.2. So, the method is global convergence.

Exercise 4

(a) Code of Polark-Ribiere conjugate gradient(PRCG) method is as below:

```
function [xMin, fMin, nIter, info] = descentLineSearch(F, descent, ls, alpha0, x0, tol, maxIter)
% Parameters
% Stopping condition {'step', 'grad'}
stopType = 'grad';
% Initialization
nIter = 0;
x_k = x0;
p_k=-F.df(x0);
alpha_k=alpha0;
info.xs = x0;
info.alphas = alpha0;
stopCond = false;
% Loop until convergence or maximum number of iterations
while (~stopCond && nIter <= maxIter)
  % Increment iterations
    nIter = nIter + 1;
    % Compute descent direction
    switch lower(descent)
      case 'cg'
        df_k_1=F.df(x_k);
    end
    % Call line search given by handle ls to compute step length alpha_k
    alpha_k=ls(x_k,p_k,alpha0);
    % Update x_k
    x_k_1 = x_k;
    %df_k_1=F.df(x_k_1);
    % YOUR CODE HERE
    x_k=x_k+alpha_k*p_k;
    df_k=F.df(x_k);
    beta_k=(df_k'*(df_k-df_k_1))/(df_k_1'*df_k_1);
    %Update p_k
    p_k=-df_k+beta_k.*p_k;
    % Store iteration info
    info.xs = [info.xs x_k];
    info.alphas = [info.alphas alpha_k];
    % Stopping conditions
    switch stopType
      case 'step'
        % Compute relative step length
        normStep = norm(x_k - x_k_1)/norm(x_k_1);
        stopCond = (normStep < tol);
      case 'grad'
        stopCond = (norm(F.df(x_k), 'inf') < tol*(1 + abs(F.f(x_k))));
    end
end
% Assign output values
xMin = x_k;
fMin = F.f(x k);
```

(b) We apply PR-CG method for two initial points (-1,2) and (-1, -0.25). In this exercise, we use backtracking with ρ=0.1, c_1=10^{-4} as line search method to adapt step length, and set stopping tolerance as ε=10^{-4}. So, we get the iteration plot as below.

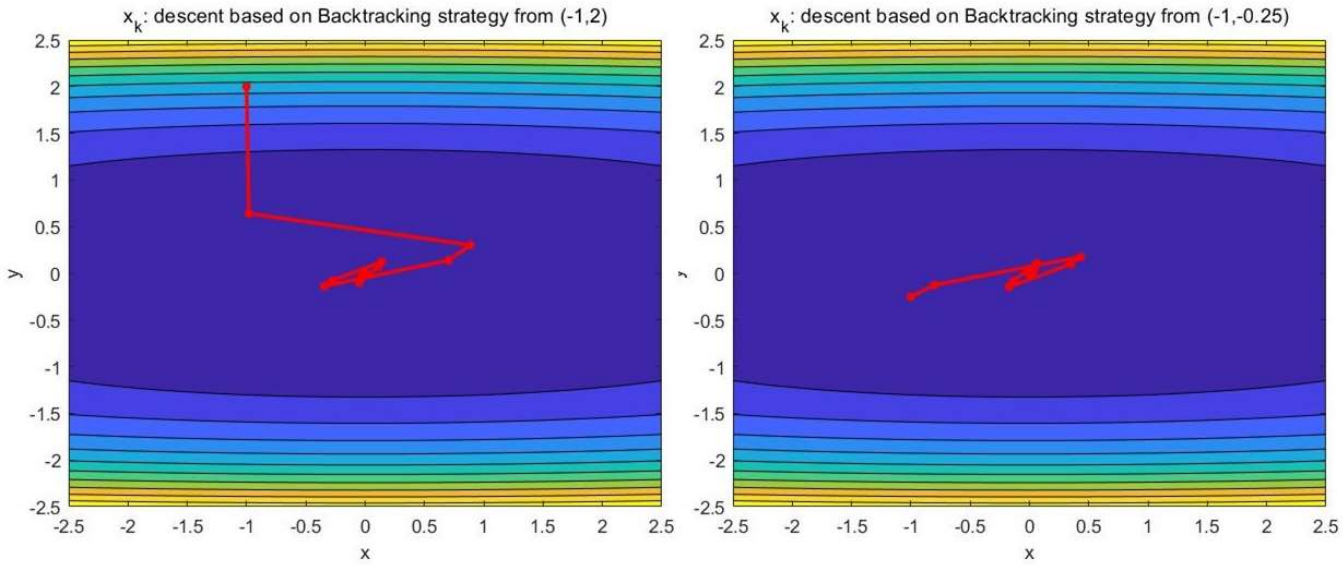


Fig 8. Iteration plot based on PRCG method

(c) From Fig 8, we find that sometimes the step length is too large and the searching direction is not always descent direction. The reason is that the objective function is not a quadratic optimization problem. In this case, the conjugate gradient method does not usually converge to a minimal point within n steps. As the iterations proceed, the search direction will no longer be the conjugate direction of Hessian matrix.

(d) Global convergence is not guaranteed in this case. The conjugate gradient depends highly on the line search method, the conjugacy and descent property of p_k and p_{k+1} cannot be guaranteed by the method mentioned in (a). So, we try to modify our PRCG solver by utilizing strong Wolfe condition (c_1=10^{-4}, c_2=0.1) as the line search method, and apply a simple adaptation $\beta_{k+1} = \max\{\beta_{k+1}^{PR}, 0\}$ to ensure the descent property.

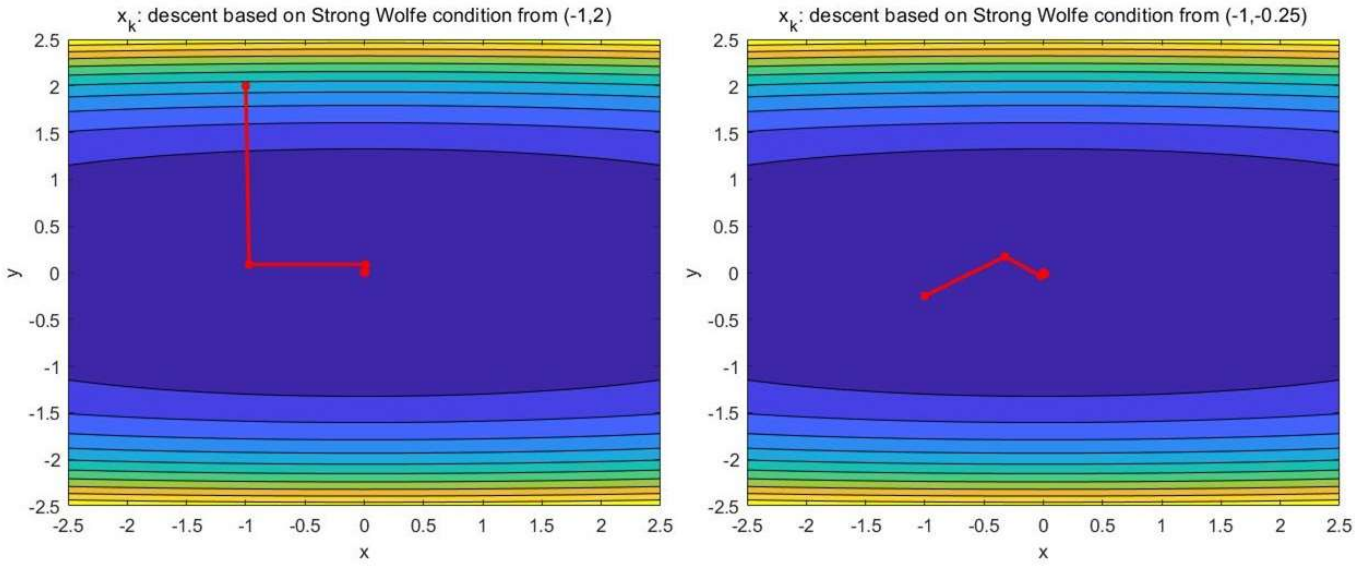
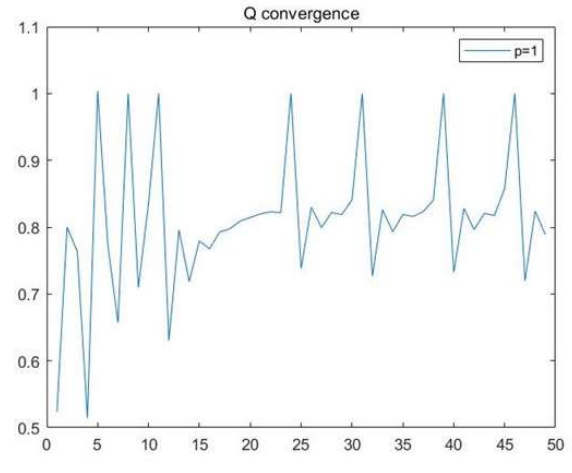


Fig 9. Iteration plot based on modified PRCG method

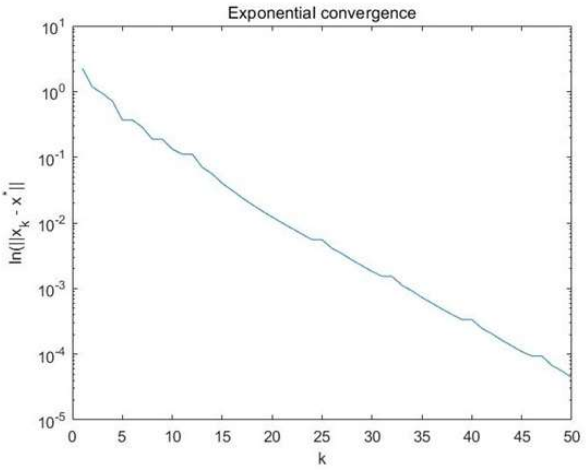
Comparing Figure 8 and Figure 9, it is clear that iteration converges to the minimizer efficiently and the numerical minimizer is closer to the posterior minimizer. Theoretically, strongly convex property of the objective function guarantees: (1) The level set $L = \{x: f(x) \leqslant f(x_0)\}$ be bounded. (2) f is Lipschitz continuously differentiable. Additionally, the modified CG is implemented with line search satisfying strong Wolfe conditions and decent searching direction. Hence, the global convergence is guaranteed by these

conditions.

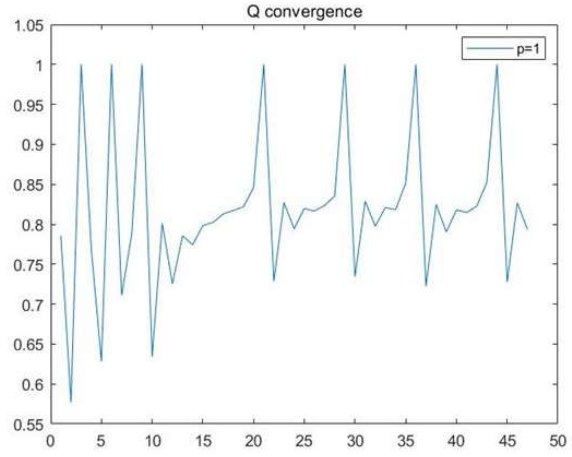
(e) We plot the relevant error plot for PRCG method in (b)



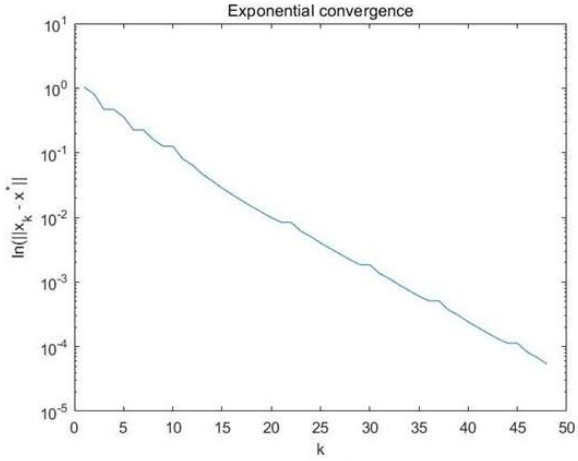
(i)



(ii)



(iii)



(iv)

Fig 10. Convergence plot for PRCG: (i) (ii) for starting point (-1,2); (iii) (iv) for starting point (-1,-0.25)

From Figure 10, we can predict that the convergence order is linear from (i) (iii), and we can predict the convergence rate from the slope of the fitting line in (ii) (iv), which is 0.0084. In theory, the convergence situation of CG submits to $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \leqslant \frac{\sqrt{\|\nabla^2 f\|_2\|\nabla^2 f^{-1}\|_2}-1}{\sqrt{\|\nabla^2 f\|_2\|\nabla^2 f^{-1}\|_2}+1}$, indicating linear convergence with rate 0.1716, which agrees with the result of empirical result.

Exercise 5

(a) The implementation of BFGS is as follows:

```
function [xMin, fMin, nIter, info] = descentLineSearchBFGS(F, descent, ls, alpha0, x0, tol, maxIter)
% Parameters
% Stopping condition {'step', 'grad'}
stopType = 'grad';
% Initialization
nIter = 0;
x_k = x0;
H_k=eye(2);
p_k=-F.df(x0);
alpha_k=alpha0;
info.xs = x0;
info.alphas = alpha0;
stopCond = false;
% Loop until convergence or maximum number of iterations
while (~stopCond && nIter <= maxIter)
  % Increment iterations
    nIter = nIter + 1;
    % Compute descent direction
    switch lower(descent)
     case 'bfgs'
        p_k=-H_k*F.df(x_k);
    end
    % Call line search given by handle ls to compute step length alpha_k
    alpha_k=ls(x_k,p_k,alpha0);
    % Update x_k
    x_k_1 = x_k;
    x_k=x_k+alpha_k*p_k;
    s_k=x_k-x_k_1;
    y_k=F.df(x_k)-F.df(x_k_1);
    if nIter==1
        H_k=((s_k'*y_k)/(y_k'*y_k))*H_k;
        info.H=H_k;
    end
    H_k_1=H_k;
    H_k=(eye(2)-(1/(y_k'*s_k))*s_k*y_k')*H_k_1'*(eye(2)-(1/(y_k'*s_k))*y_k*s_k')+(1/(y_k'*s_k))*(s_k*s_k');
    % Store iteration info
    info.xs = [info.xs x_k];
    info.alphas = [info.alphas alpha_k];
    info.H=[info.H H_k];
    % Stopping conditions
    switch stopType
      case 'step'
        % Compute relative step length
        normStep = norm(x_k - x_k_1)/norm(x_k_1);
        stopCond = (normStep < tol);
      case 'grad'
        stopCond = (norm(F.df(x_k), 'inf') < tol*(1 + abs(F.f(x_k))));
    end
end
% Assign output values
xMin = x_k;
fMin = F.f(x_k);
```

This implement could be efficient because we use the iterative formula to update the inverse of the Hessian, rather than calculate the inverse directly. Therefore, no $O(n^3)$ operations like linear system solves in our implementation, and the complexity of each iteration is around $O(n^2)$.

(b) We apply the BFGS implementation to the objective function $f(x,y)=(x-3y)^2+x^4$, starting from (10,10). We set initial step length as 10/9, stopping tolerance as 10^{-4} and use the Wolfe conditions with $c_1=10^{-4}$, $c_2=0.1$ for line search because we want to ensure the gradients are sampled at x_k allowing the approximation model m_k to obtain adequate curvature information.

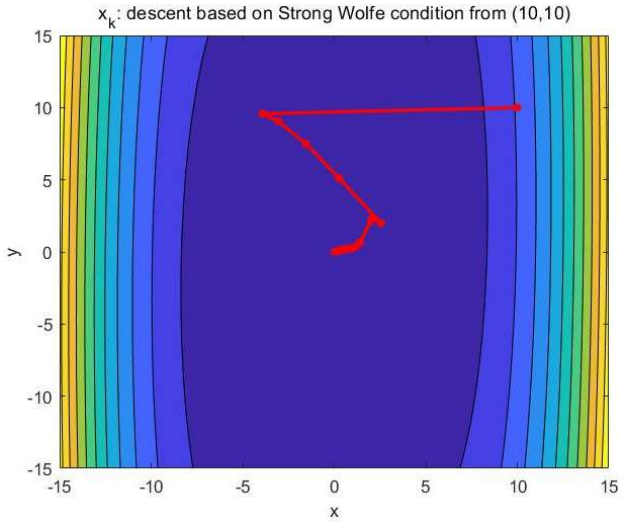


Fig 11. Iteration plot for BFGS

We find that the iteration converges to the minimizer effectively and no zig-zag characteristic is shown.

(c)

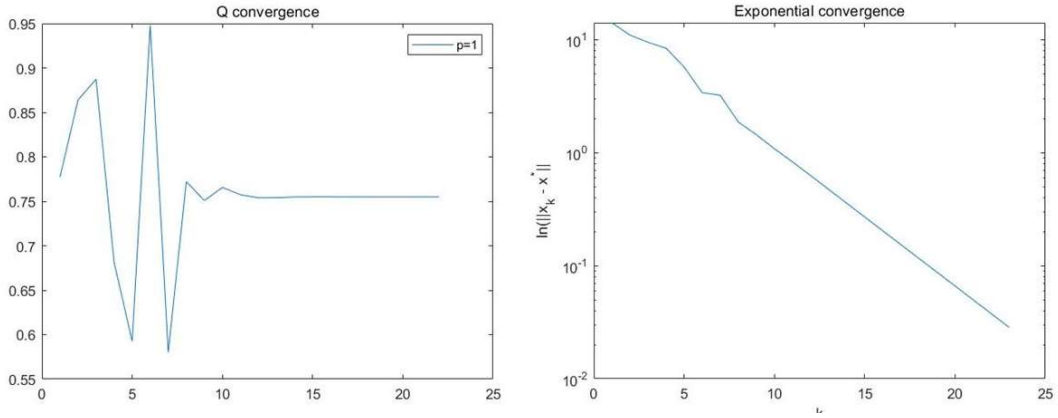


Fig 12. Convergence plot for BFGS

From Fig 12, we predict that this model is linear convergence with rate around 0.75, because we can find that $\frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|} \leqslant 0.75$, when k is sufficient large and the slope of linear fitting curve of exponential convergence is about 0.75. Theoretically, f is twice differentiable continuously and in $N(x^*,\varepsilon)$ (neighborhood for x^*), $\nabla^2 f(x^*)$ is S.P.D and $\nabla^2 f(x)$ is locally Lipschitz continuous(with L=12ε). BFGS method is at least linear convergence under this case, which is corresponding to the convergence reflected by plot.

(d) BFGS is global convergence, because of the conditions as follows:

(1) f *is* twice continuously differentiable and the level set of x_0=(10,10) $L=\{x\in\mathbb{R}^n : f(x)\leqslant f(x_0)\}$ is convex.

(2) Since $\nabla^2 f(x)\succ 0$, $\forall x\in L$, then $\exists\ m>0, \forall z\in R^2,\ m\|z\|^2\leqslant z^T\nabla^2 f(x)z$

(3) L is bounded, then $\exists\ M>m>0, \forall z\in R^2,\ z^T\nabla^2 f(x)z\leqslant M\|z\|^2$.

Hence, BFGS iteration converges to the unique minimizer x^*.

(e)

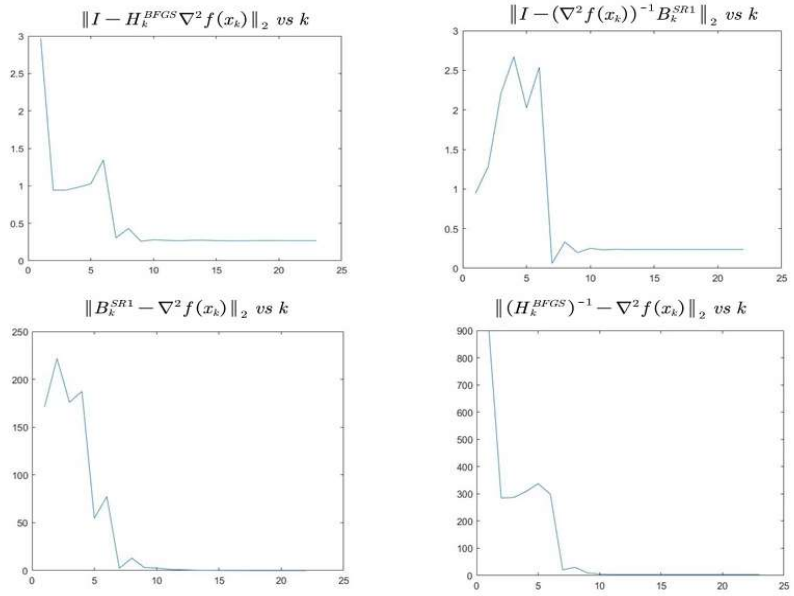


Fig 13. Hessian approximation by two methods

From the plot, we find that the SR-1 Hessian approximation is superior to BFGS. Because when *k* is sufficient large, we find $\left\|\mathrm{B}_k^{SR1} - \nabla^2\,\mathrm{f}(\mathrm{x}_k)\right\|_2 \ll \left\|(\mathrm{H}_k^{BFGS})^{-1} - \nabla^2\,\mathrm{f}(\mathrm{x}_k)\right\|_2$ and $\left\|I - H_k^{SR1}\nabla^2\,\mathrm{f}(\mathrm{x}_k)\right\|_2 \approx \left\|\mathrm{I} - (\nabla^2\,\mathrm{f}(\mathrm{x}_k)^{-1})(\mathrm{B}_k^{SR1})^{-1}\right\|_2$. So, it seems that the B_k^{SR} is more efficient in approximating $\nabla^2\,\mathrm{f}(\mathrm{x}_k)$. In theory, *f* is twice continuously differentiable with the Hessian bounded and Lipschitz continuous in a neighborhood of x^*. Additionally, the steps s_k are uniformly independent and $\left|(y_k - B_k s_k)^T s_k\right| \geq r\|s_k\|\,\|y_k - B_k s_k\|$ holds for all k. Hence, B_k generated by SR1 satisfy $\lim_{k\to\infty}\|B_k - \nabla^2 f(x^*)\| = 0$, indicating that B_k approximates the Hessian matrix of x^* to a high extend.