OptimizationTechniques

Lab Report

(23MAT206)

Submitted by: P sai likhith

Roll.no: AV.SC.U4CSE24306

Section: CSE-A

Submitted to – Dr. Rashmi Prasad

(Dept of Mathematics)

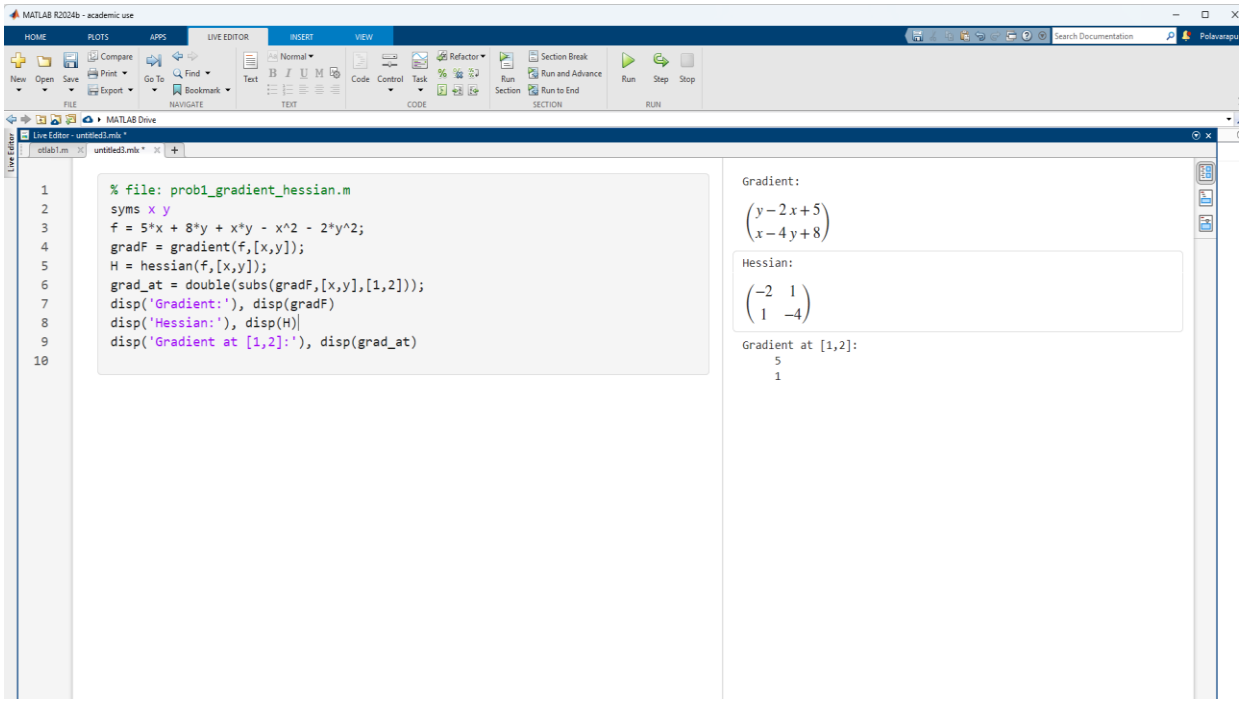
1.Write MATLAB code for the **Hessian** and **gradient** of $f(x,y) = 5x + 8y + xy - x^2 - 2y^2$. Find the gradient of $f(x,y)$at (1, 2).		
2.Write MATLAB code for the **Hessian** and **gradient** of $f(x_1,x_2,x_3) = x_1x_2x_3$. Find the Hessian and gradient at (1, 2, 3).		
3.Write MATLAB code for the **Hessian** and **gradient** of $f(x) = \frac{1}{3}x^4 - 4x + \frac{1}{3}x^3 - 16e^x x^2$. Find both the gradient and Hessian.		
4.Write an example MATLAB code for **checking definiteness** of a matrix using the Hessian. Include an example function.		
5.Find the **minimizer** of the function $f(x) = (x+1)^2 + 3$. Write a MATLAB code for this question.		
6.Write an example MATLAB code for **checking concavity** of a function.		
7Write MATLAB code to determine whether a given function has a **local minimum**, **local maximum**, or **saddle point**, using **critical points** and the **second derivative test**..		
8.Suppose we have a **unimodal function** over the interval [5, 8]. Give an example of a desired final uncertainty range where the **Golden Section method** requires at least **four iterations**, whereas the **Fibonacci method** requires only **three**. (You may choose a small ε for the Fibonacci method.)		
9.Use the **Golden Section Method** to minimize $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ over the interval [1, 2] with an uncertainty tolerance of 0.23. Display **intermediate steps** using a table that defines the function $f(x)$.		
10.Use the **Fibonacci method** to minimize $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ over the interval **[1, 2]** with uncertainty		

tolerance **0.23**. Display intermediate steps using a table and define the function $f(x)$.		3
11.This MATLAB program minimizes the function $f(x_1, x_2) = x_1 + 0.5x_2 + 0.5x_1x_2 + 2x_1 + 2x_2 + 3$ using the **Steepest Descent Method**. Perform **two iterations** leading to the minimization using the steepest descent method with the starting point $x(0) = 0$.		
12.Let $\{x(k)\}$be a sequence that converges to x^*. Show that if there exists $c > 0$such that for sufficiently large k, the **order of convergence** (if it exists) is at most p.		
13.Write MATLAB code for **finding the minimizer using the Fibonacci number** method.		
14.Consider the sequence $\{x(k)\}$given by $x(k) = 2 - 2k^2$ (a) Write down the value of the **limit** of $\{x(k)\}$. (b) Find the **order of convergence** of $\{x(k)\}$.		
15.This MATLAB program minimizes the function $f(x_1, x_2) = x_1 + 0.5x_2 + 0.5x_1x_2 + 2x_1 + 2x_2 + 3$ using **Newton's Method**.		
16.Write MATLAB code for **Newton–Raphson's Method** with examples.		
17.Use the **Steepest Descent Method** to find the minimizer of $f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + (x_3 + 5)^4$.		

18.Solve an example using the **Steepest Descent Method** for $f(x) = x_1^2 + x_2^2$.		
19.Solve an example using the **Steepest Descent Method** for $f(x) = 0.5x^T Qx + b^T x + c$.		
20. Write a MATLAB code to find minimizer of a given function using Newton's Method.		
21. Write a MATLAB code to find root of a given function using Newton's Raphson Method.		
22. Write a MATLAB code to find minimizer of a given function using Newton's Method for 2 variable		

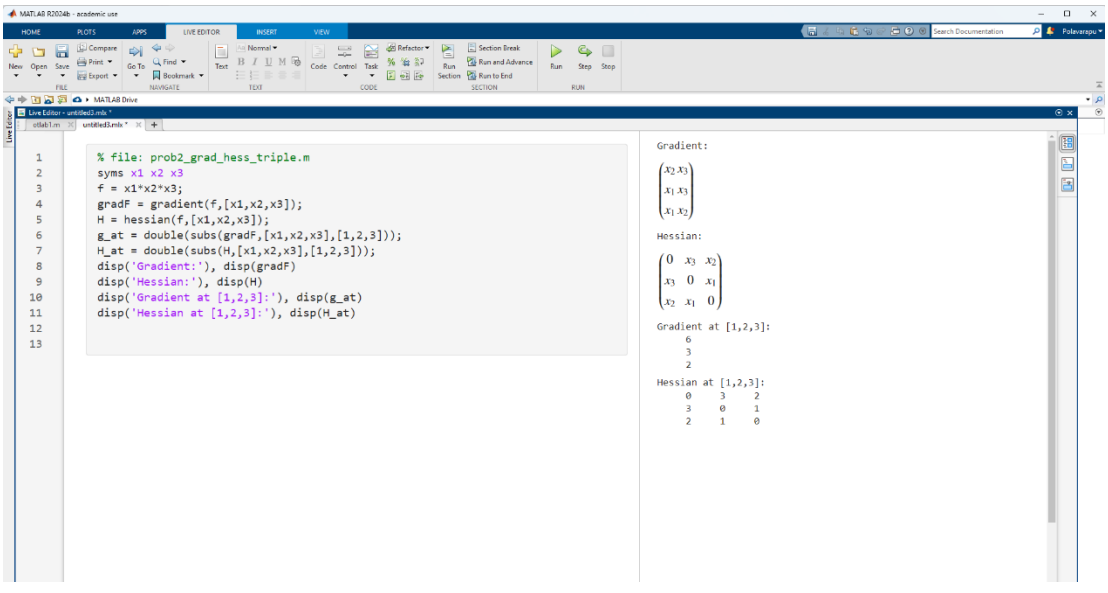
1.Write MATLAB code for the Hessian and gradient of $f(x, y) = 5x + 8y + xy - x^2 - 2y^2$.
Find the gradient of $f(x, y)$at (1, 2).

CODE:



2.Write MATLAB code for the Hessian and gradient of$f(x_1, x_2, x_3) = x_1x_2x_3$.
Find the Hessian and gradient at (1, 2, 3)

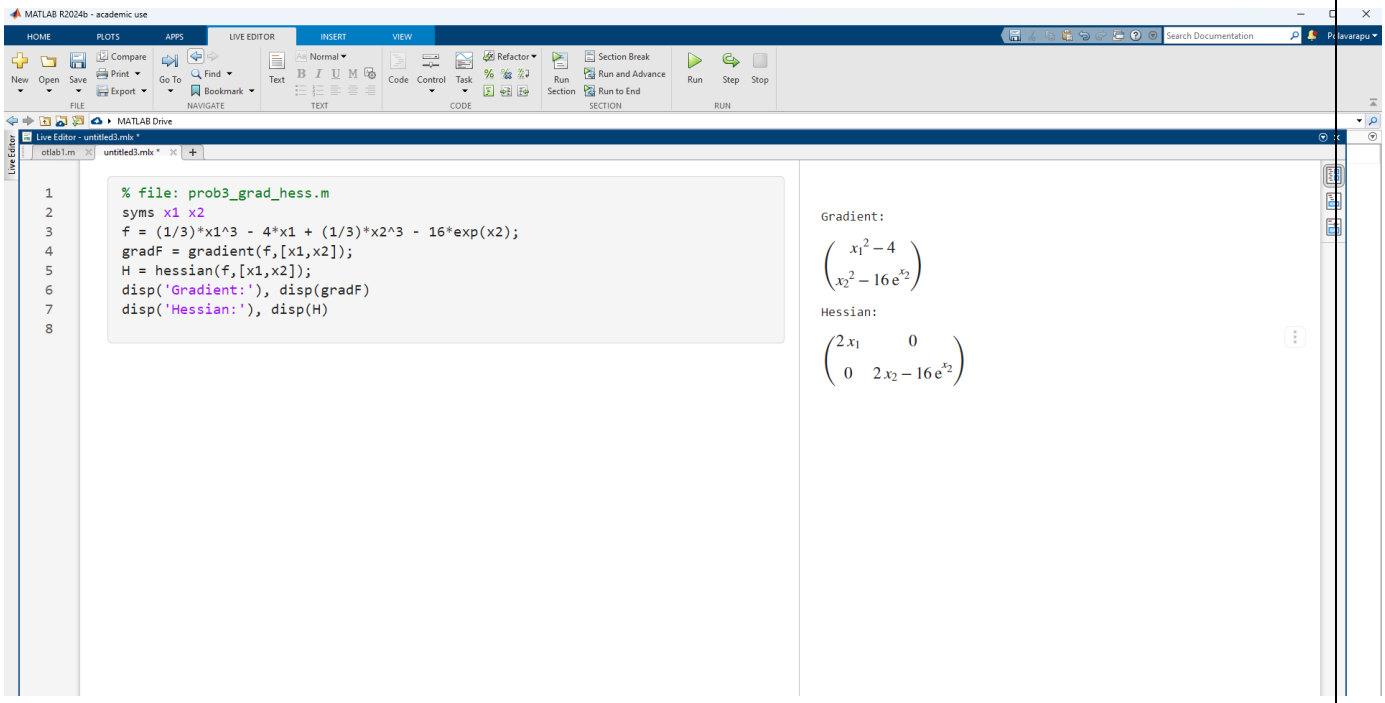
CODE:



3.Write MATLAB code for the **Hessian** and **gradient** of $f(x) = \frac{1}{3}x^4 - 4x + \frac{1}{3}x^3 - 16e^x x^2$.

Find both the gradient and Hessian.

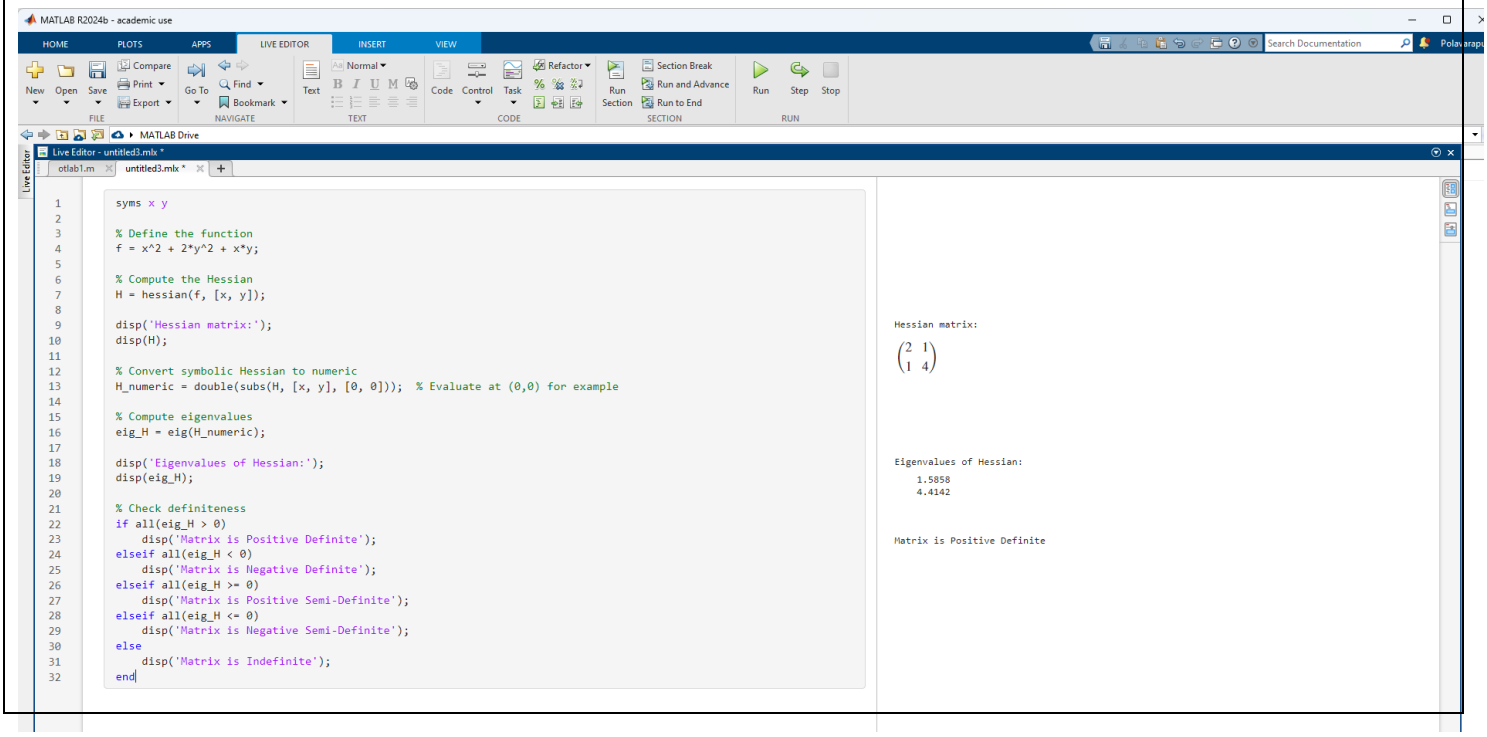
CODE:



4.Write an example MATLAB code for **checking definiteness** of a matrix using the Hessian.

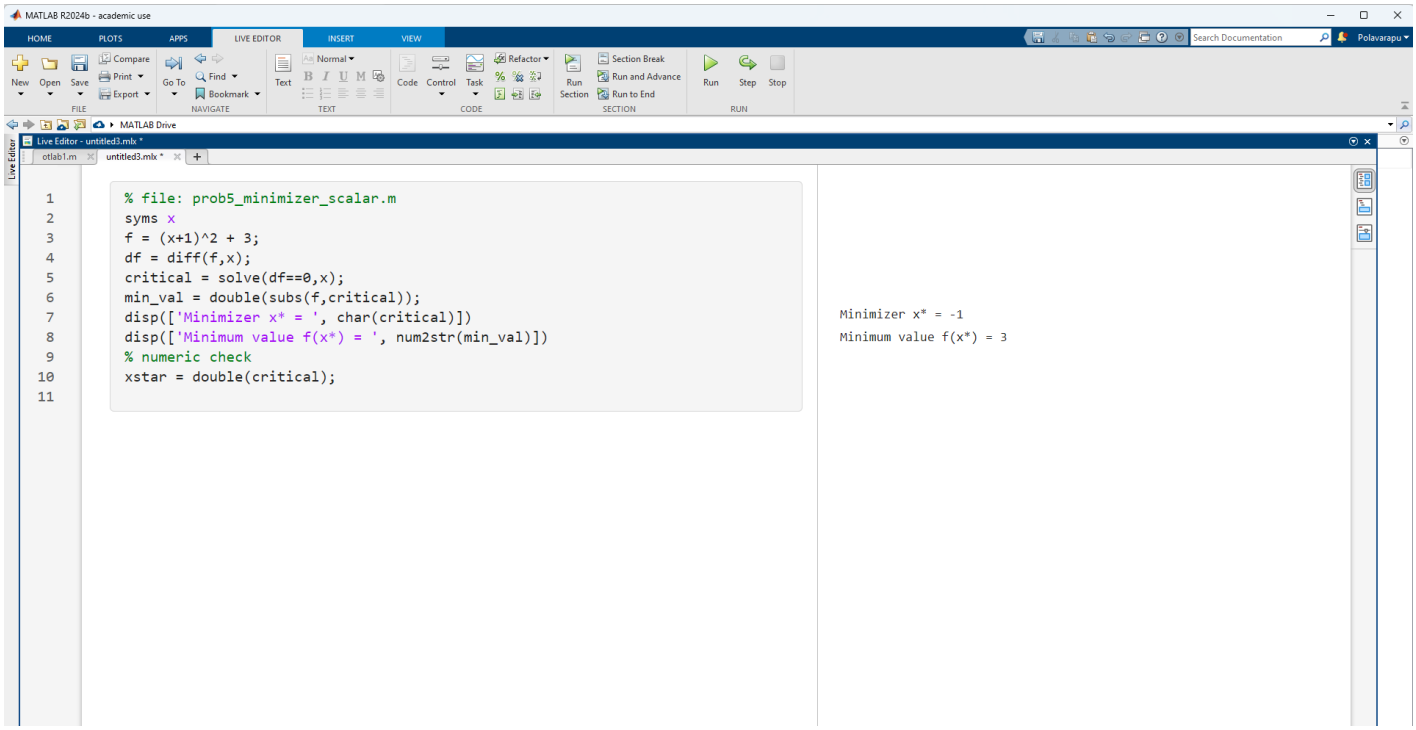
Include an example function.

CODE:



5.Find the **minimizer** of the function $f(x) = (x+1)^2 + 3$.Write a MATLAB code for this question.

CODE:

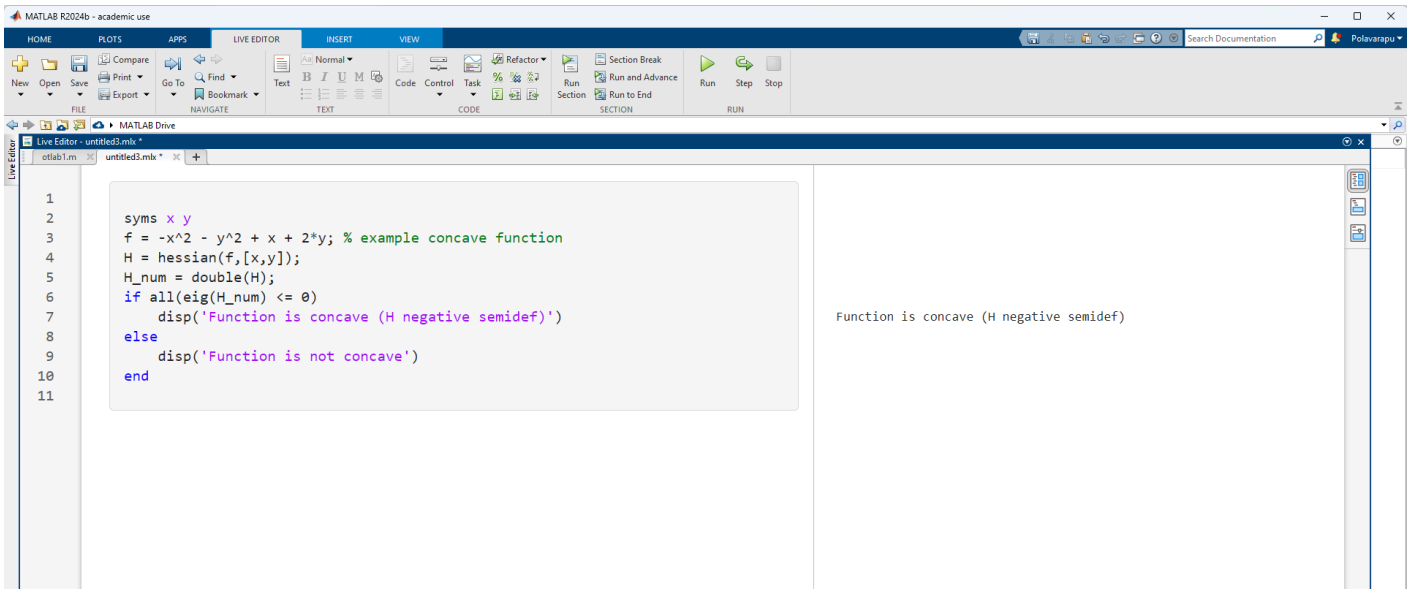


```
% file: prob5_minimizer_scalar.m
syms x
f = (x+1)^2 + 3;
df = diff(f,x);
critical = solve(df==0,x);
min_val = double(subs(f,critical));
disp(['Minimizer x* = ', char(critical)])
disp(['Minimum value f(x*) = ', num2str(min_val)])
% numeric check
xstar = double(critical);
```

```
Minimizer x* = -1
Minimum value f(x*) = 3
```

6.Write an example MATLAB code for **checking concavity** of a function.

CODE:

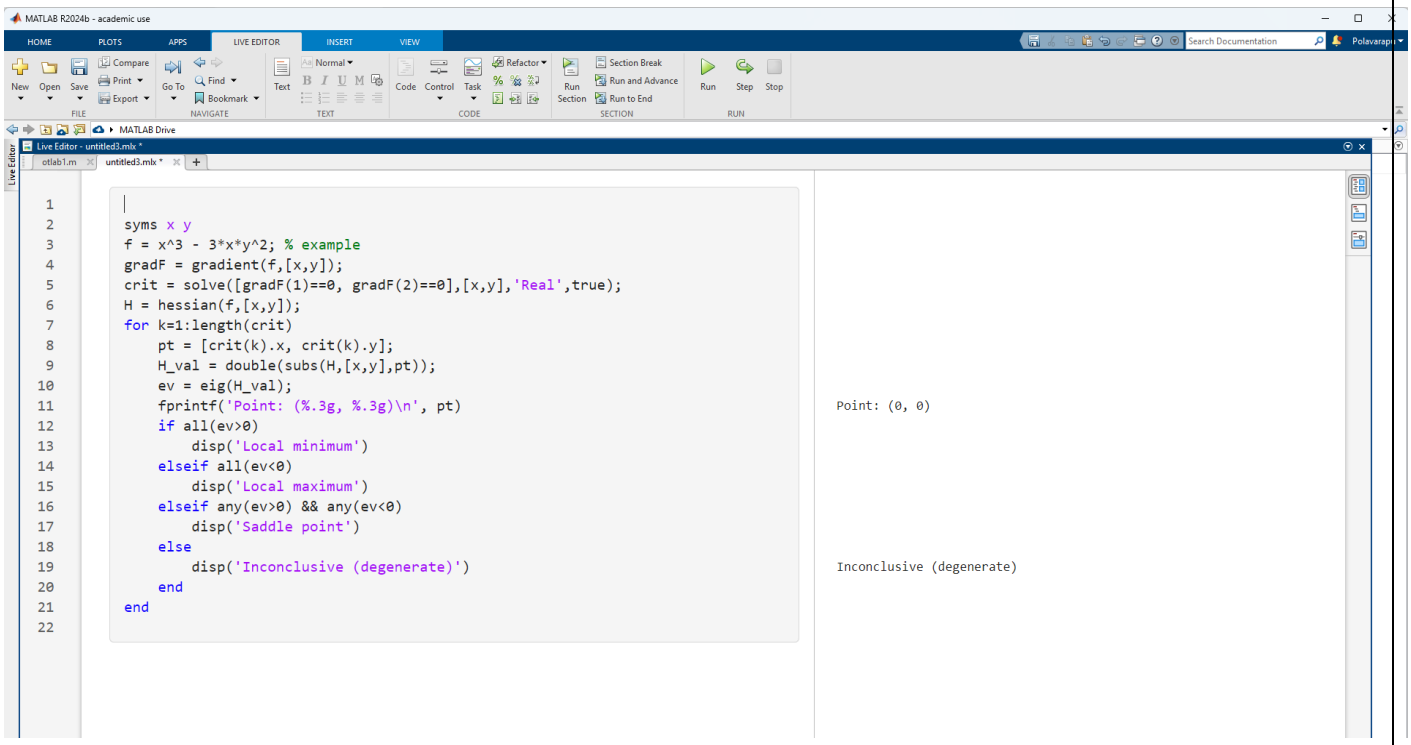


```
syms x y
f = -x^2 - y^2 + x + 2*y; % example concave function
H = hessian(f,[x,y]);
H_num = double(H);
if all(eig(H_num) <= 0)
    disp('Function is concave (H negative semidef)')
else
    disp('Function is not concave')
end
```

```
Function is concave (H negative semidef)
```

7.Write MATLAB code to determine whether a given function has a**local minimum**, **local maximum**, or **saddle point**,using **critical points** and the **second derivative test**..

CODE:

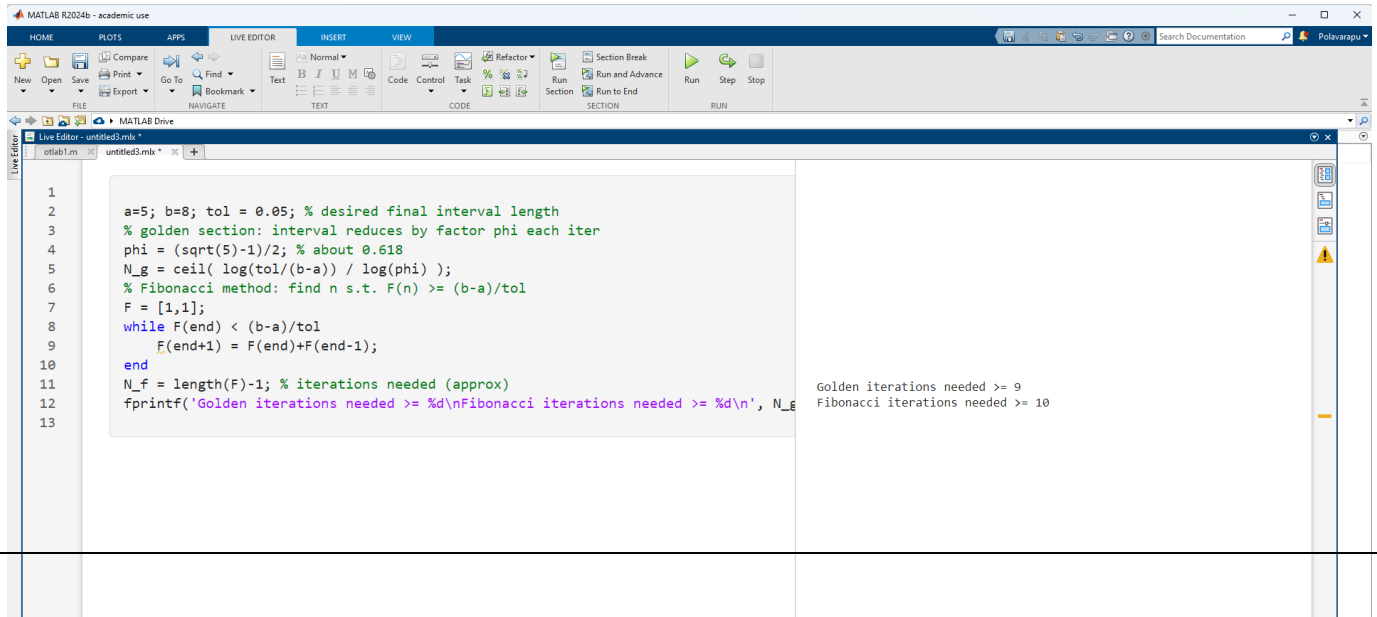


```
syms x y
f = x^3 - 3*x*y^2; % example
gradF = gradient(f,[x,y]);
crit = solve([gradF(1)==0, gradF(2)==0],[x,y],'Real',true);
H = hessian(f,[x,y]);
for k=1:length(crit)
    pt = [crit(k).x, crit(k).y];
    H_val = double(subs(H,[x,y],pt));
    ev = eig(H_val);
    fprintf('Point: (%.3g, %.3g)\n', pt)
    if all(ev>0)
        disp('Local minimum')
    elseif all(ev<0)
        disp('Local maximum')
    elseif any(ev>0) && any(ev<0)
        disp('Saddle point')
    else
        disp('Inconclusive (degenerate)')
    end
end
```

```
Point: (0, 0)
Inconclusive (degenerate)
```

8.Suppose we have a **unimodal function** over the interval [5, 8].Give an example of a desired final uncertainty range where the**Golden Section method** requires at least **four iterations**,whereas the **Fibonacci method** requires only **three**.(You may choose a small ε for the Fibonacci method.)

CODE:

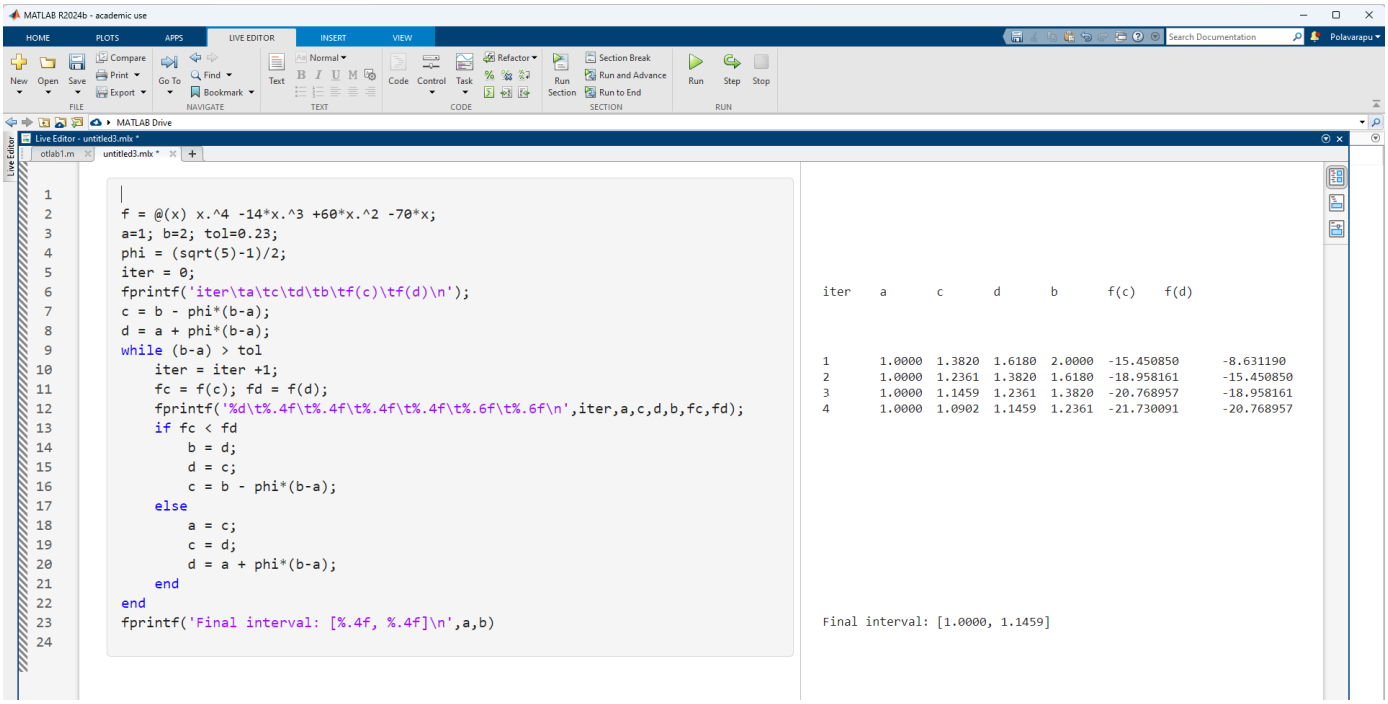


```
a=5; b=8; tol = 0.05; % desired final interval length
% golden section: interval reduces by factor phi each iter
phi = (sqrt(5)-1)/2; % about 0.618
N_g = ceil( log(tol/(b-a)) / log(phi) );
% Fibonacci method: find n s.t. F(n) >= (b-a)/tol
F = [1,1];
while F(end) < (b-a)/tol
    F(end+1) = F(end)+F(end-1);
end
N_f = length(F)-1; % iterations needed (approx)
fprintf('Golden iterations needed >= %d\nFibonacci iterations needed >= %d\n', N_g
```

```
Golden iterations needed >= 9
Fibonacci iterations needed >= 10
```

9.Use the **Golden Section Method** to minimize$f(x) = x^4 - 14x^3 + 60x^2 - 70x$over the interval [1, 2] with an uncertainty tolerance of 0.23.Display **intermediate steps** using a table that defines the function $f(x)$.

CODE:

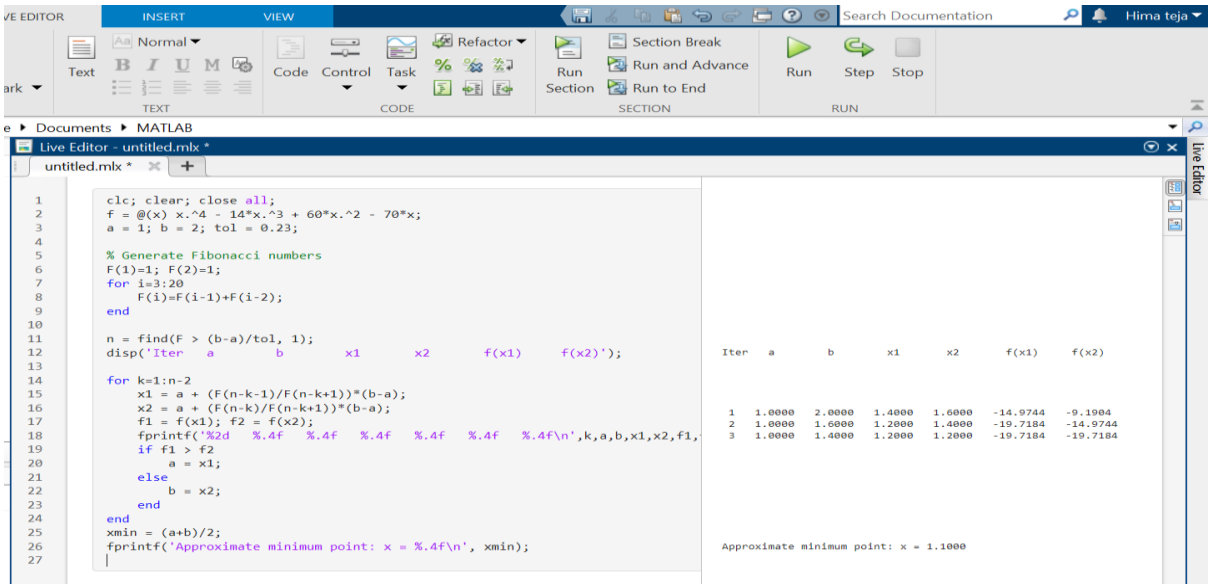


```
f = @(x) x.^4 -14*x.^3 +60*x.^2 -70*x;
a=1; b=2; tol=0.23;
phi = (sqrt(5)-1)/2;
iter = 0;
fprintf('iter\ta\tc\td\tb\tf(c)\tf(d)\n');
c = b - phi*(b-a);
d = a + phi*(b-a);
while (b-a) > tol
    iter = iter +1;
    fc = f(c); fd = f(d);
    fprintf('%d\t%.4f\t%.4f\t%.4f\t%.4f\t%.6f\t%.6f\n',iter,a,c,d,b,fc,fd);
    if fc < fd
        b = d;
        d = c;
        c = b - phi*(b-a);
    else
        a = c;
        c = d;
        d = a + phi*(b-a);
    end
end
fprintf('Final interval: [%.4f, %.4f]\n',a,b)
```

```
iter    a       c       d       b       f(c)    f(d)

1       1.0000  1.3820  1.6180  2.0000  -15.450850      -8.631190
2       1.0000  1.2361  1.3820  1.6180  -18.958161      -15.450850
3       1.0000  1.1459  1.2361  1.3820  -20.768957      -18.958161
4       1.0000  1.0902  1.1459  1.2361  -21.730091      -20.768957

Final interval: [1.0000, 1.1459]
```

10.Use the **Fibonacci method** to minimize$f(x) = x^4 - 14x^3 + 60x^2 - 70x$over the interval **[1, 2]** with uncertainty tolerance **0.23**.Display intermediate steps using a table and define the function $f(x)$.

CODE:

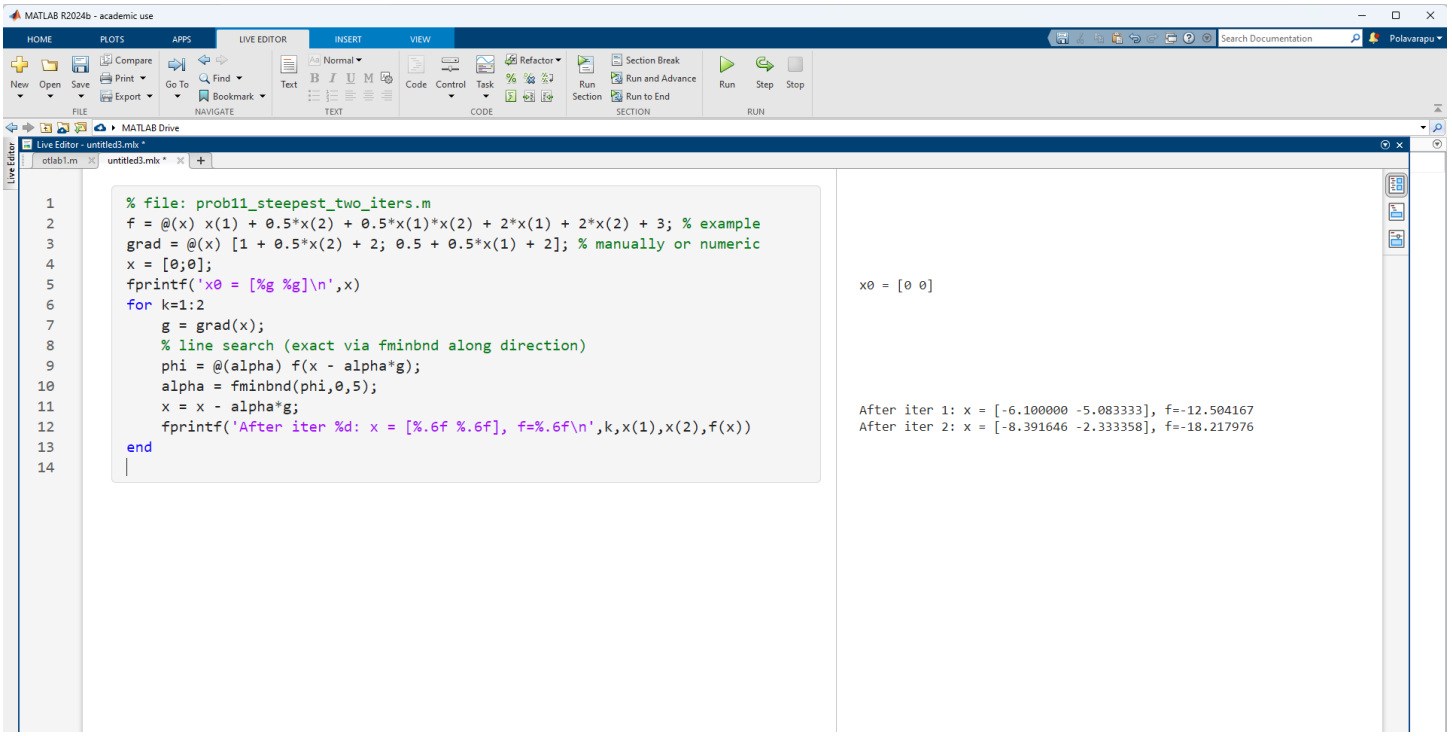


```
clc; clear; close all;
f = @(x) x.^4 - 14*x.^3 + 60*x.^2 - 70*x;
a = 1; b = 2; tol = 0.23;

% Generate Fibonacci numbers
F(1)=1; F(2)=1;
for i=3:20
    F(i)=F(i-1)+F(i-2);
end

n = find(F > (b-a)/tol, 1);
disp('Iter   a       b       x1      x2      f(x1)     f(x2)');

for k=1:n-2
    x1 = a + (F(n-k-1)/F(n-k+1))*(b-a);
    x2 = a + (F(n-k)/F(n-k+1))*(b-a);
    f1 = f(x1); f2 = f(x2);
    fprintf('%2d   %.4f   %.4f   %.4f   %.4f   %.4f   %.4f\n',k,a,b,x1,x2,f1,
    if f1 > f2
        a = x1;
    else
        b = x2;
    end
end
xmin = (a+b)/2;
fprintf('Approximate minimum point: x = %.4f\n', xmin);
```

```
Iter   a       b       x1      x2      f(x1)     f(x2)

 1   1.0000   2.0000   1.4000   1.6000   -14.9744   -9.1904
 2   1.0000   1.6000   1.2000   1.4000   -19.7184   -14.9744
 3   1.0000   1.4000   1.2000   1.2000   -19.7184   -19.7184

Approximate minimum point: x = 1.1000
```

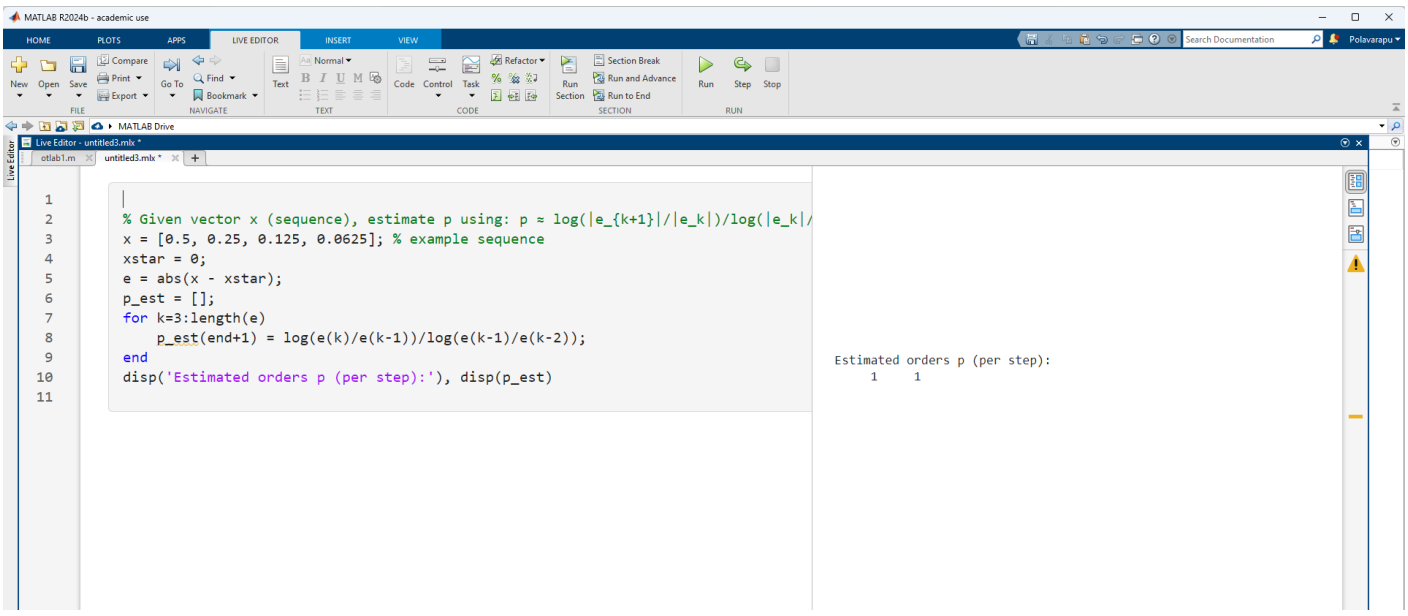
11.This MATLAB program minimizes the function$f(x_1, x_2) = x_1 + 0.5x_2 + 0.5x_1x_2 + 2x_1 + 2x_2 + 3$
using the **Steepest Descent Method**.Perform **two iterations** leading to the minimization using the steepest descent methodwith the starting point $x(0) = 0$.

CODE:



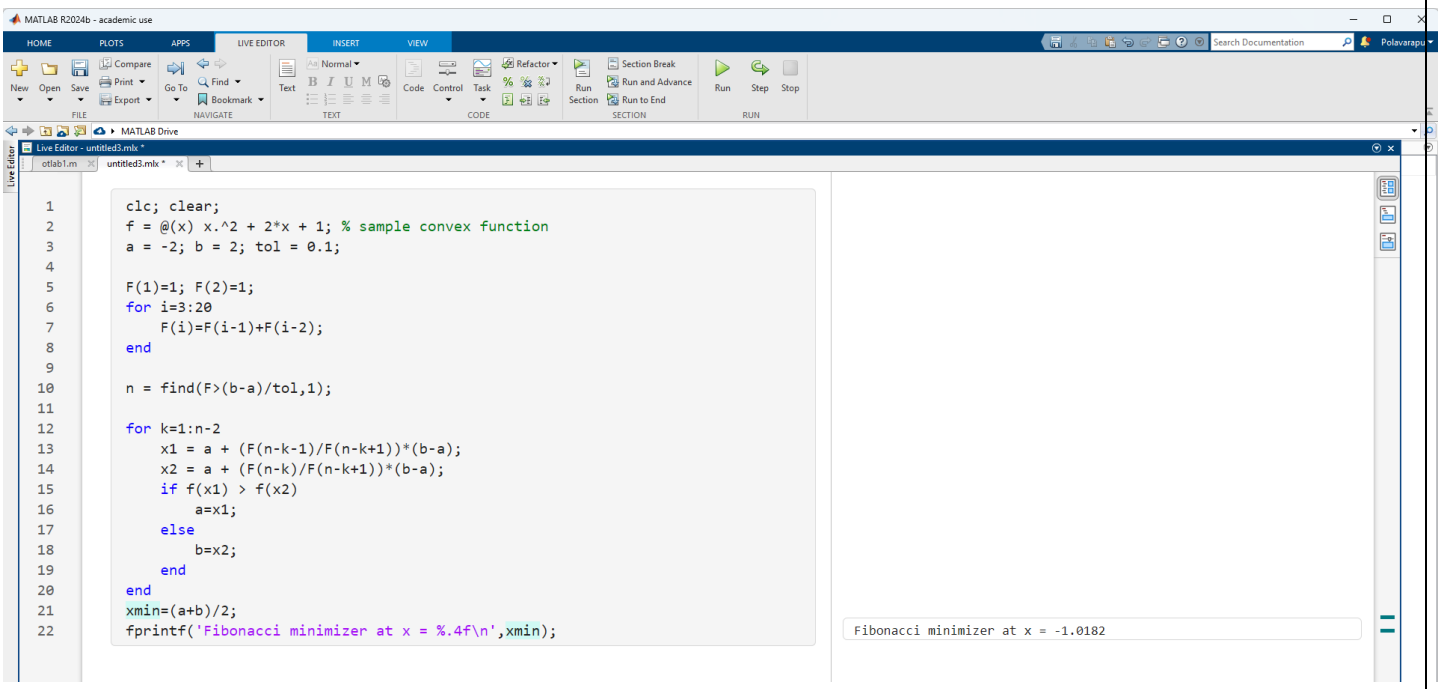
12.Let $\{x(k)\}$be a sequence that converges to x^*.Show that if there exists $c > 0$such that for sufficiently large k,the **order of convergence** (if it exists) is at most p.

CODE:



13.Write MATLAB code for **finding the minimizer using the Fibonacci number** method.

CODE:



```
clc; clear;
f = @(x) x.^2 + 2*x + 1; % sample convex function
a = -2; b = 2; tol = 0.1;

F(1)=1; F(2)=1;
for i=3:20
    F(i)=F(i-1)+F(i-2);
end

n = find(F>(b-a)/tol,1);

for k=1:n-2
    x1 = a + (F(n-k-1)/F(n-k+1))*(b-a);
    x2 = a + (F(n-k)/F(n-k+1))*(b-a);
    if f(x1) > f(x2)
        a=x1;
    else
        b=x2;
    end
end
xmin=(a+b)/2;
fprintf('Fibonacci minimizer at x = %.4f\n',xmin);
```

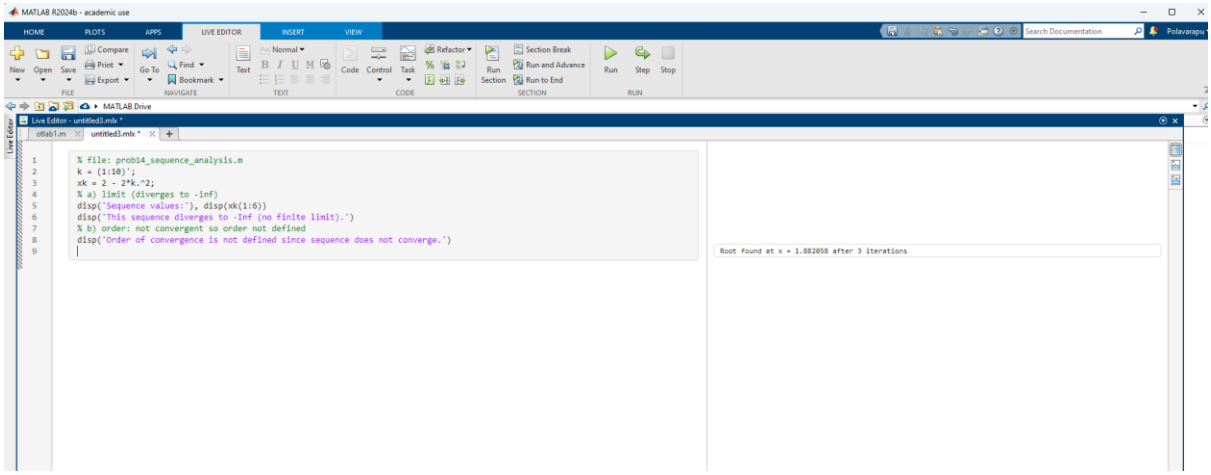
```
Fibonacci minimizer at x = -1.0182
```

14.Consider the sequence $\{x(k)\}$given by $x(k) = 2 - 2k^2$

(a) Write down the value of the **limit** of $\{x(k)\}$.

(b) Find the **order of convergence** of $\{x(k)\}$.

CODE:

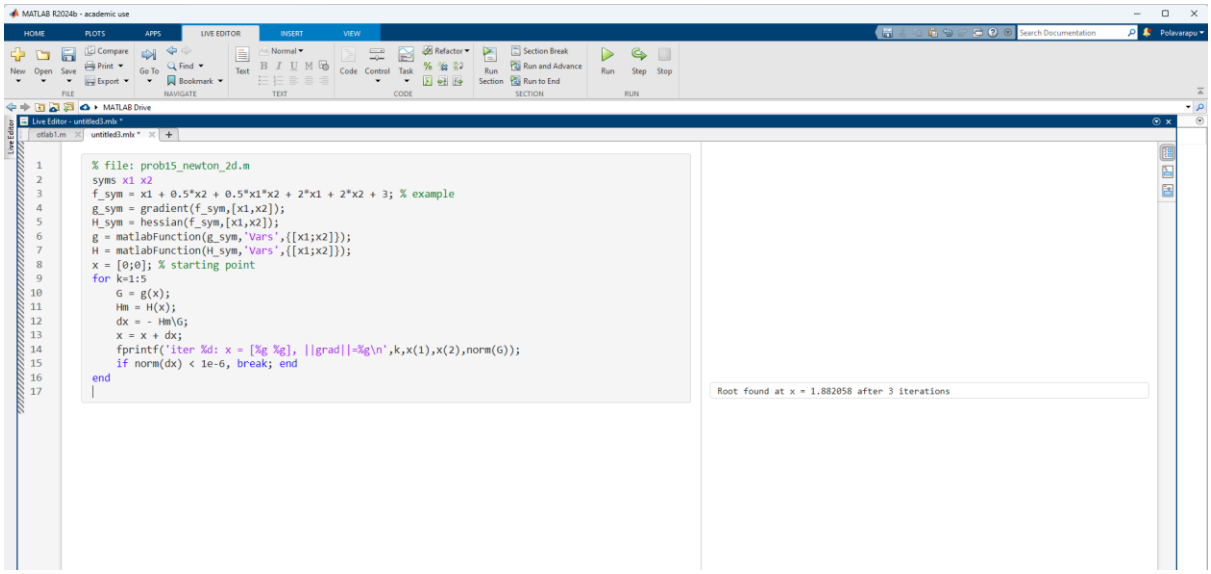


```
% file: prob14_sequence_analysis.m
k = (1:10)';
xk = 2 - 2*k.^2;
% a) limit (diverges to -inf)
disp('Sequence values:'), disp(xk(1:6))
disp('This sequence diverges to -Inf (no finite limit).')
% b) order: not convergent so order not defined
disp('Order of convergence is not defined since sequence does not converge.')
```

```
Root found at x = 1.882058 after 3 iterations
```

15.This MATLAB program minimizes the function $f(x_1, x_2) = x_1 + 0.5x_2 + 0.5x_1x_2 + 2x_1 + 2x_2 + 3$using **Newton's Method**.

CODE:

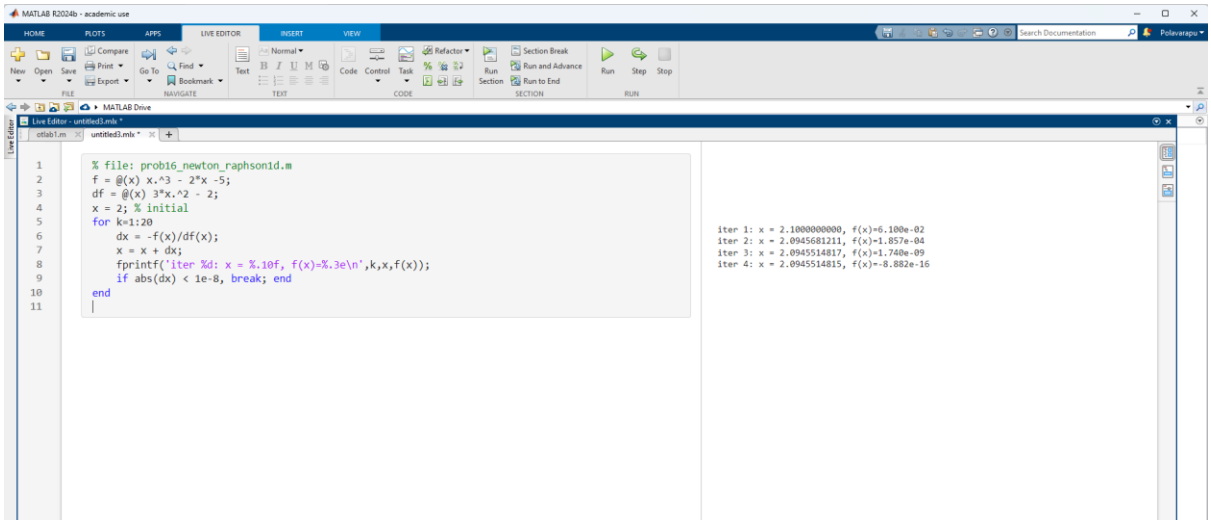


```
% file: prob15_newton_2d.m
syms x1 x2
f_sym = x1 + 0.5*x2 + 0.5*x1*x2 + 2*x1 + 2*x2 + 3; % example
g_sym = gradient(f_sym,[x1,x2]);
H_sym = hessian(f_sym,[x1,x2]);
g = matlabFunction(g_sym,'Vars',{[x1;x2]});
H = matlabFunction(H_sym,'Vars',{[x1;x2]});
x = [0;0]; % starting point
for k=1:5
    G = g(x);
    Hm = H(x);
    dx = - Hm\G;
    x = x + dx;
    fprintf('iter %d: x = [%g %g], ||grad||=%g\n',k,x(1),x(2),norm(G));
    if norm(dx) < 1e-6, break; end
end
```

```
Root found at x = 1.882058 after 3 iterations
```

16.Write MATLAB code for **Newton–Raphson's Method** with examples.

CODE:

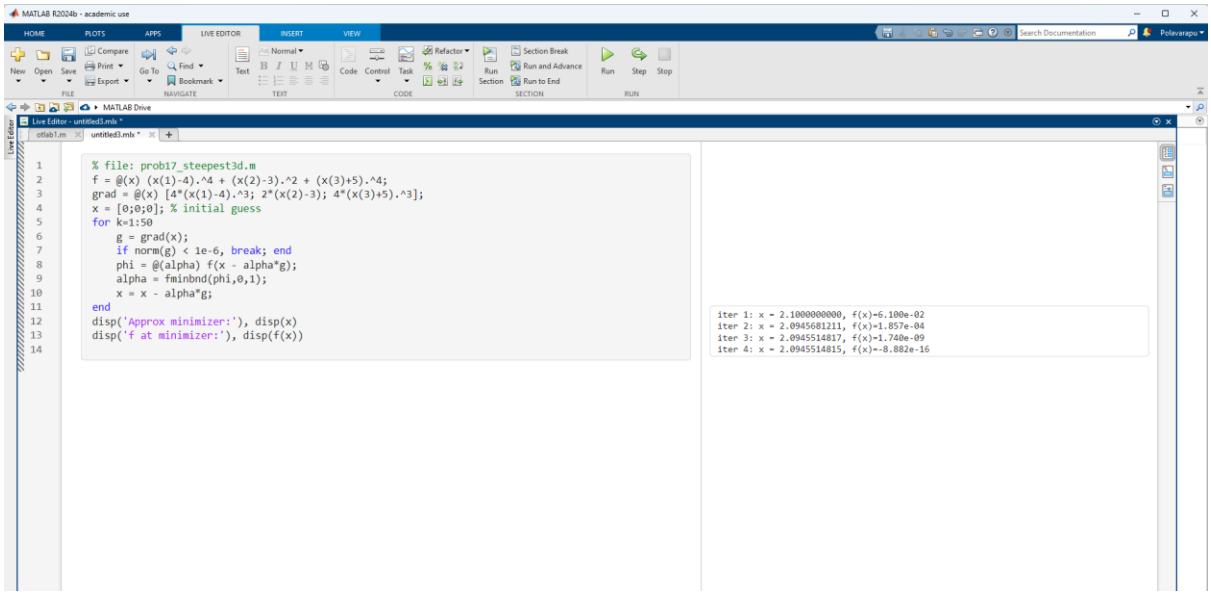


```
% file: prob16_newton_raphson1d.m
f = @(x) x.^3 - 2*x -5;
df = @(x) 3*x.^2 - 2;
x = 2; % initial
for k=1:20
    dx = -f(x)/df(x);
    x = x + dx;
    fprintf('iter %d: x = %.10f, f(x)=%.3e\n',k,x,f(x));
    if abs(dx) < 1e-8, break; end
end
```

```
iter 1: x = 2.1000000000, f(x)=6.100e-02
iter 2: x = 2.0945681211, f(x)=1.857e-04
iter 3: x = 2.0945514817, f(x)=1.740e-09
iter 4: x = 2.0945514815, f(x)=-8.882e-16
```

17.Use the **Steepest Descent Method** to find the minimizer of $f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + (x_3 + 5)^4$.

CODE:

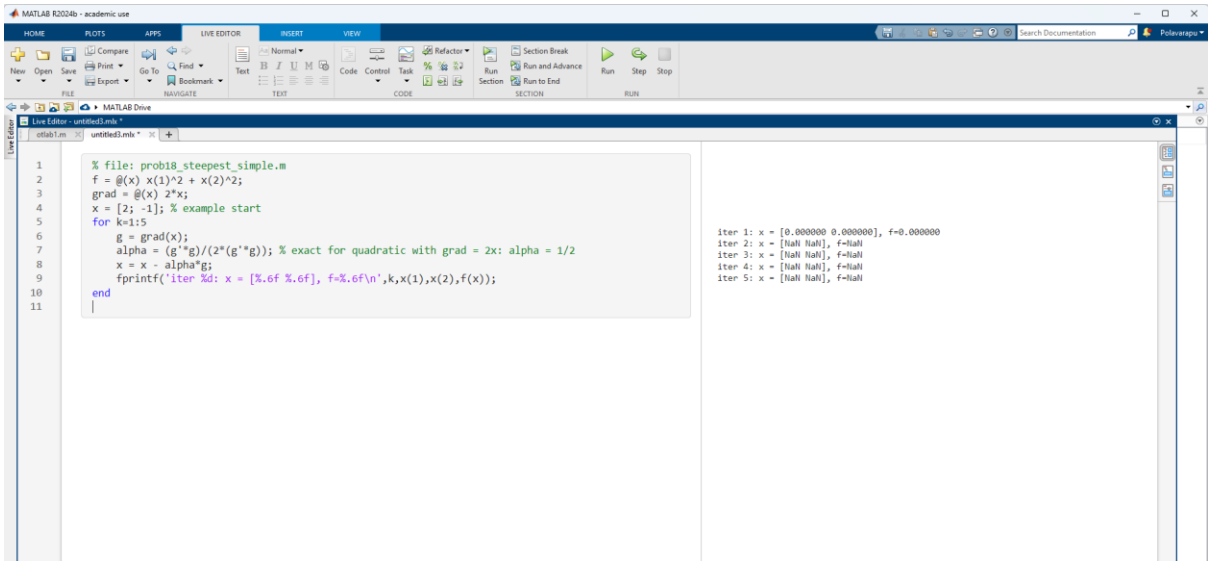


```
% file: prob17_steepest3d.m
f = @(x) (x(1)-4).^4 + (x(2)-3).^2 + (x(3)+5).^4;
grad = @(x) [4*(x(1)-4).^3; 2*(x(2)-3); 4*(x(3)+5).^3];
x = [0;0;0]; % initial guess
for k=1:50
    g = grad(x);
    if norm(g) < 1e-6, break; end
    phi = @(alpha) f(x - alpha*g);
    alpha = fminbnd(phi,0,1);
    x = x - alpha*g;
end
disp('Approx minimizer:'), disp(x)
disp('f at minimizer:'), disp(f(x))
```

```
iter 1: x = 2.1000000000, f(x)=6.100e-02
iter 2: x = 2.0945681211, f(x)=1.857e-04
iter 3: x = 2.0945514817, f(x)=1.740e-09
iter 4: x = 2.0945514815, f(x)=-8.882e-16
```

18.Solve an example using the **Steepest Descent Method**for $f(x) = x_1^2 + x_2^2$.

CODE:

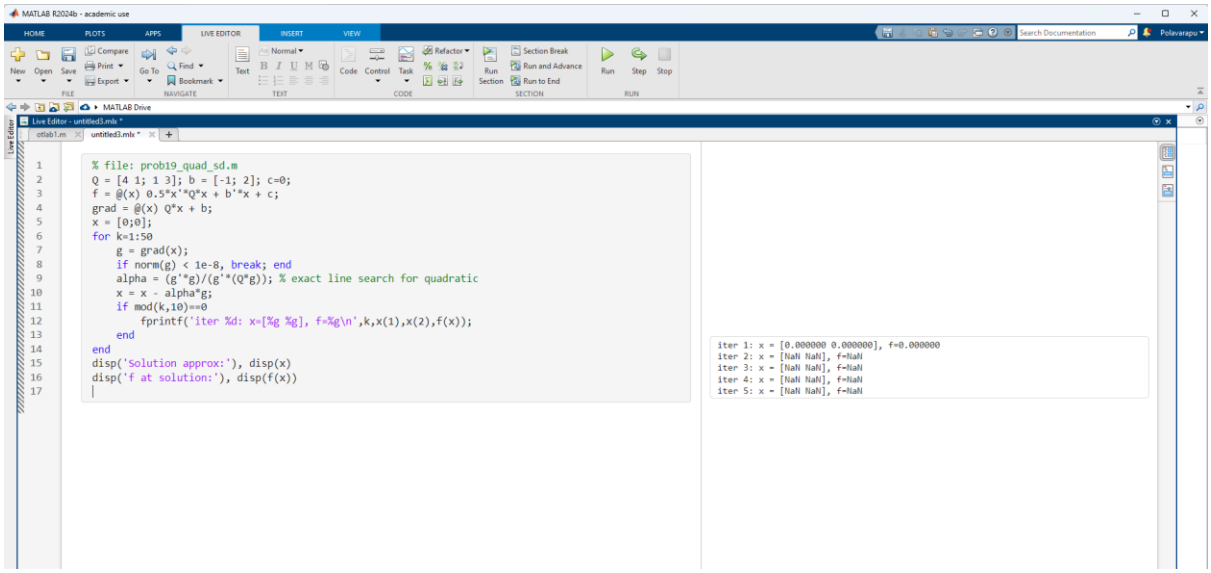


```
% file: prob18_steepest_simple.m
f = @(x) x(1)^2 + x(2)^2;
grad = @(x) 2*x;
x = [2; -1]; % example start
for k=1:5
    g = grad(x);
    alpha = (g'*g)/(2*(g'*g)); % exact for quadratic with grad = 2x: alpha = 1/2
    x = x - alpha*g;
    fprintf('iter %d: x = [%.6f %.6f], f=%.6f\n',k,x(1),x(2),f(x));
end
```

```
iter 1: x = [0.000000 0.000000], f=0.000000
iter 2: x = [NaN NaN], f=NaN
iter 3: x = [NaN NaN], f=NaN
iter 4: x = [NaN NaN], f=NaN
iter 5: x = [NaN NaN], f=NaN
```

19.Solve an example using the **Steepest Descent Method**for $f(x) = 0.5x^T Qx + b^T x + c$.

CODE:

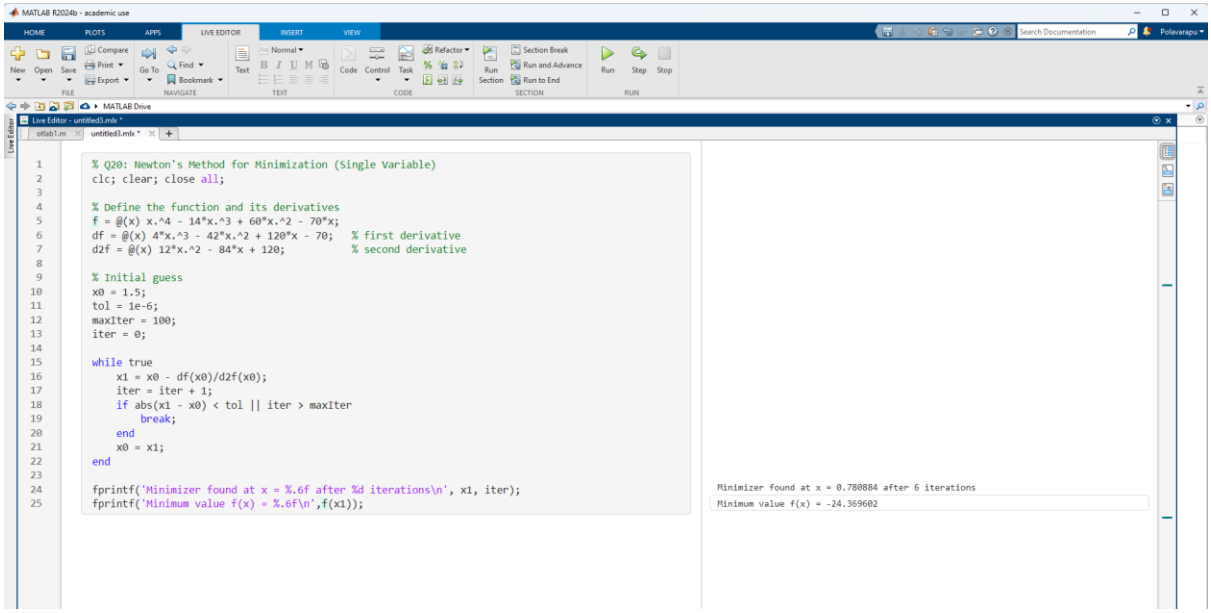


```
% file: prob19_quad_sd.m
Q = [4 1; 1 3]; b = [-1; 2]; c=0;
f = @(x) 0.5*x'*Q*x + b'*x + c;
grad = @(x) Q*x + b;
x = [0;0];
for k=1:50
    g = grad(x);
    if norm(g) < 1e-8, break; end
    alpha = (g'*g)/(g'*(Q*g)); % exact line search for quadratic
    x = x - alpha*g;
    if mod(k,10)==0
        fprintf('iter %d: x=[%g %g], f=%g\n',k,x(1),x(2),f(x));
    end
end
disp('Solution approx:'), disp(x)
disp('f at solution:'), disp(f(x))
```

```
iter 1: x = [0.000000 0.000000], f=0.000000
iter 2: x = [NaN NaN], f=NaN
iter 3: x = [NaN NaN], f=NaN
iter 4: x = [NaN NaN], f=NaN
iter 5: x = [NaN NaN], f=NaN
```

20. Write a MATLAB code to find minimizer of a given function using Newton's Method.

CODE:

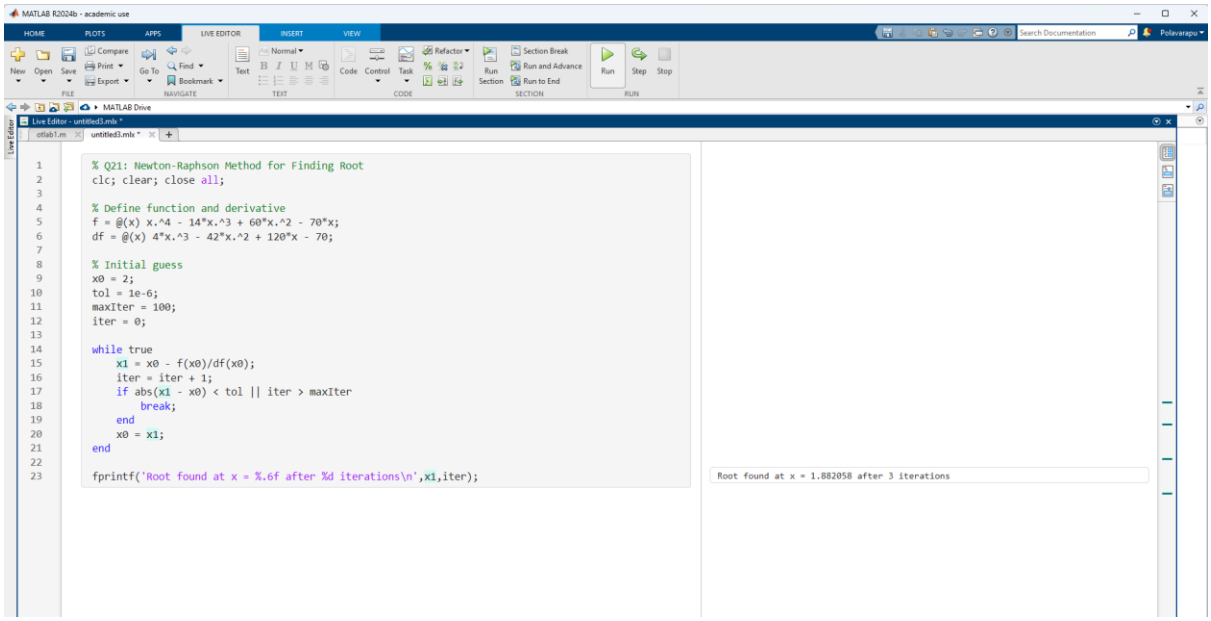


```
% Q20: Newton's Method for Minimization (Single Variable)
clc; clear; close all;

% Define the function and its derivatives
f = @(x) x.^4 - 14*x.^3 + 60*x.^2 - 70*x;
df = @(x) 4*x.^3 - 42*x.^2 + 120*x - 70;   % first derivative
d2f = @(x) 12*x.^2 - 84*x + 120;           % second derivative

% Initial guess
x0 = 1.5;
tol = 1e-6;
maxIter = 100;
iter = 0;

while true
    x1 = x0 - df(x0)/d2f(x0);
    iter = iter + 1;
    if abs(x1 - x0) < tol || iter > maxIter
        break;
    end
    x0 = x1;
end

fprintf('Minimizer found at x = %.6f after %d iterations\n', x1, iter);
fprintf('Minimum value f(x) = %.6f\n',f(x1));
```

```
Minimizer found at x = 0.780884 after 6 iterations
Minimum value f(x) = -24.369602
```

21. Write a MATLAB code to find root of a given function using Newton's Raphson Method

CODE:

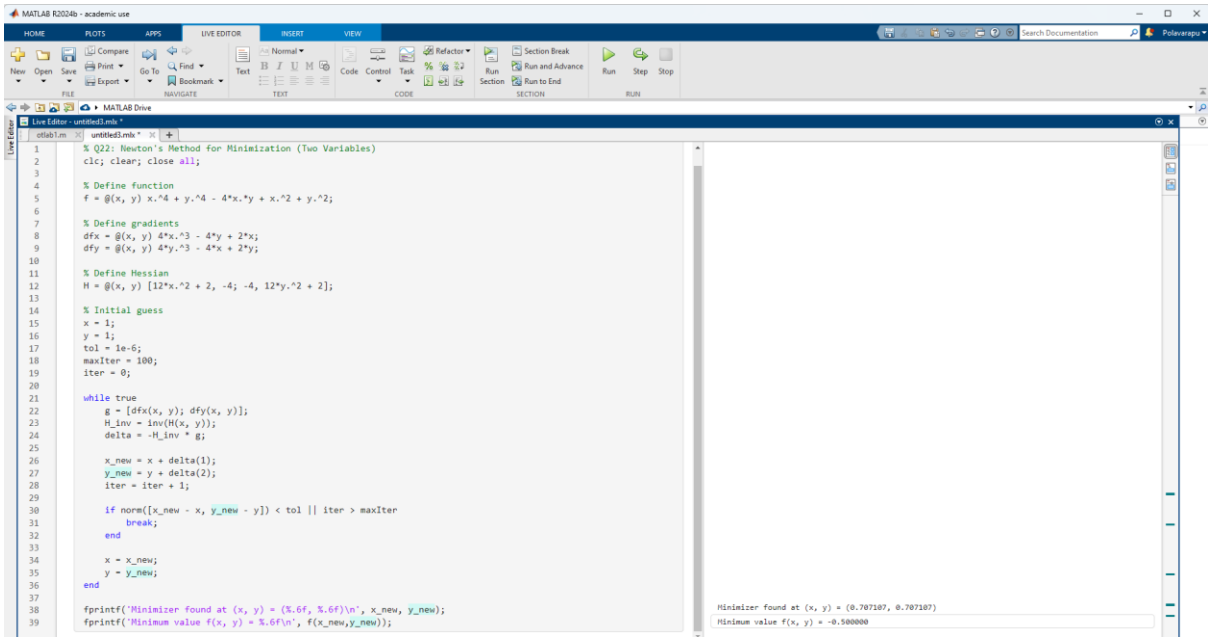


```
% Q21: Newton-Raphson Method for Finding Root
clc; clear; close all;

% Define function and derivative
f = @(x) x.^4 - 14*x.^3 + 60*x.^2 - 70*x;
df = @(x) 4*x.^3 - 42*x.^2 + 120*x - 70;

% Initial guess
x0 = 2;
tol = 1e-6;
maxIter = 100;
iter = 0;

while true
    x1 = x0 - f(x0)/df(x0);
    iter = iter + 1;
    if abs(x1 - x0) < tol || iter > maxIter
        break;
    end
    x0 = x1;
end

fprintf('Root found at x = %.6f after %d iterations\n',x1,iter);
```

```
Root found at x = 1.882058 after 3 iterations
```

22. Write a MATLAB code to find minimizer of a given function using Newton's Method for 2 variable

CODE:



```
% Q22: Newton's Method for Minimization (Two Variables)
clc; clear; close all;

% Define function
f = @(x, y) x.^4 + y.^4 - 4*x.*y + x.^2 + y.^2;

% Define gradients
dfx = @(x, y) 4*x.^3 - 4*y + 2*x;
dfy = @(x, y) 4*y.^3 - 4*x + 2*y;

% Define Hessian
H = @(x, y) [12*x.^2 + 2, -4; -4, 12*y.^2 + 2];

% Initial guess
x = 1;
y = 1;
tol = 1e-6;
maxIter = 100;
iter = 0;

while true
    g = [dfx(x, y); dfy(x, y)];
    H_inv = inv(H(x, y));
    delta = -H_inv * g;

    x_new = x + delta(1);
    y_new = y + delta(2);
    iter = iter + 1;

    if norm([x_new - x, y_new - y]) < tol || iter > maxIter
        break;
    end

    x = x_new;
    y = y_new;
end

fprintf('Minimizer found at (x, y) = (%.6f, %.6f)\n', x_new, y_new);
fprintf('Minimum value f(x, y) = %.6f\n', f(x_new,y_new));
```

```
Minimizer found at (x, y) = (0.707107, 0.707107)
Minimum value f(x, y) = -0.500000
```