QUESTION 1

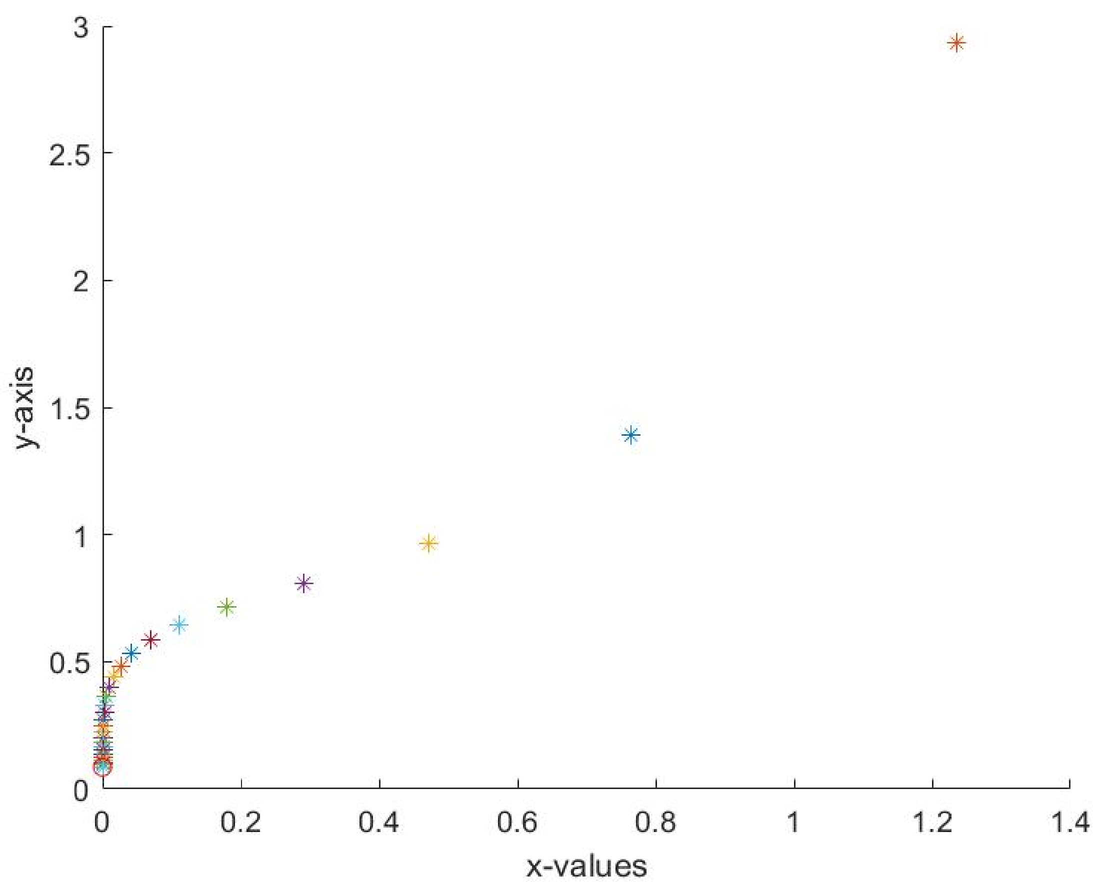
(a) GOLDEN SECTION METHOD-MINIMIZATION

```
syms x
figure; hold on;
a=0;                            % start of interval
b=2;                            % end of interval
epsilon=0.000001;               % accuracy value
iter= 50;                       % maximum number of iterations
tau=double((sqrt(5)-1)/2);      % golden proportion coefficient, around 0.618
k=0;                            % number of iterations
f = @(x) x^.2 + x.^3   % The assumed function I used
x1=a+(1-tau)*(b-a);             % computing x values
x2=a+tau*(b-a);


f_x1=f(x1);                     % computing values in x points
f_x2=f(x2);


plot(x1,f_x1,'*')              % plotting the function with x_min
plot(x2,f_x2,'*')
while ((abs(b-a)>epsilon) && (k<iter))
    k=k+1;
    if(f_x1<f_x2)
        b=x2;
        x2=x1;
        x1=a+(1-tau)*(b-a);

        f_x1=f(x1);
        f_x2=f(x2);

        plot(x1,f_x1,'*');
    else
        a=x1;
        x1=x2;
        x2=a+tau*(b-a);

        f_x1=f(x1);
        f_x2=f(x2);

        plot(x2,f_x2,'*')
    end

    k=k+1;
end
```

```
% chooses minimum point
if(f_x1<f_x2)
    sprintf('x_min=% f', x1)
    sprintf('f(x_min)=% f ', f_x1)
    plot(x1,f_x1,'ro')
else
    sprintf('x_min=% f', x2)
    sprintf('f(x_min)=% f ', f_x2)
    plot(x2,f_x2,'ro')
end
```

OUTPUT

```
 ans = 'x_min= 0.000005'
ans = 'f(x_min)= 0.085442 '
```



(b) QUADRATIC INTERPOLATION METHOD

```
clear all

f = @(x)(x^2 + x^3)  % The same function as in part A

% The x-points values

x1 = 1;

x2 = -0.5;

x3 = 2;

tol = 10^-6;


% Invoking function
quadratic_interpolation(f, x1, x2, x3, tol)
```

I created a function below

```
function quadratic_interpolation(f, x1, x2, x3, tol)
fx1 = f(x1);

    fx2 = f(x2);

    fx3 = f(x3);

    if (fx1 > fx2 && fx2 < fx3)

        cond = true;

        iterations = 1;

        while(cond)

            syms a b c

            fx1 = f(x1);

            fx2 = f(x2);

            fx3 = f(x3);

            % Quadratic interpolation in the form of a second-order polynomial
y(x) = a + bx + cx^2

            % a, b, c is calculated from the following linear equations

            eq1 = a + x1*b + x1^2 * c == fx1;
```

```
            eq2 = a + x2*b + x2^2 * c == fx2;
            eq3 = a + x3*b + x3^2 * c == fx3;
            sol = solve([eq1, eq2, eq3], [a, b, c]);
            aS = sol.a;
            bS = sol.b;
            cS = sol.c;

            xOpt = -(bS/(2*cS));
            parabolaOpt = aS + bS*xOpt + cS*(xOpt^2);
            fxOpt = f(xOpt);
            vars = [x1 x2 x3 xOpt];

            cond = (abs(fxOpt - parabolaOpt) > tol);
                if (cond)
                    iterations = iterations + 1;
                    % Calculating new points
                    vSorted = sort(vars);
                    vSIndice = find(vSorted==xOpt);
                    vSortedLeft = vSorted(:,1:(vSIndice-1));
                    vSortedRight = vSorted(:, (vSIndice+1:end));
                    x1 = max(vSortedLeft);
                    x2 = xOpt;
                    x3 = min(vSortedRight);
                end
        end
            fprintf('Optimal x* = %.4f \n', x2)
            fprintf('No. of iterations = %d \n', iterations)
    else
        disp('f(x1) > f(x2) && f(x2) < f(x3) not satisfied')
    end
end
```

OUTPUT

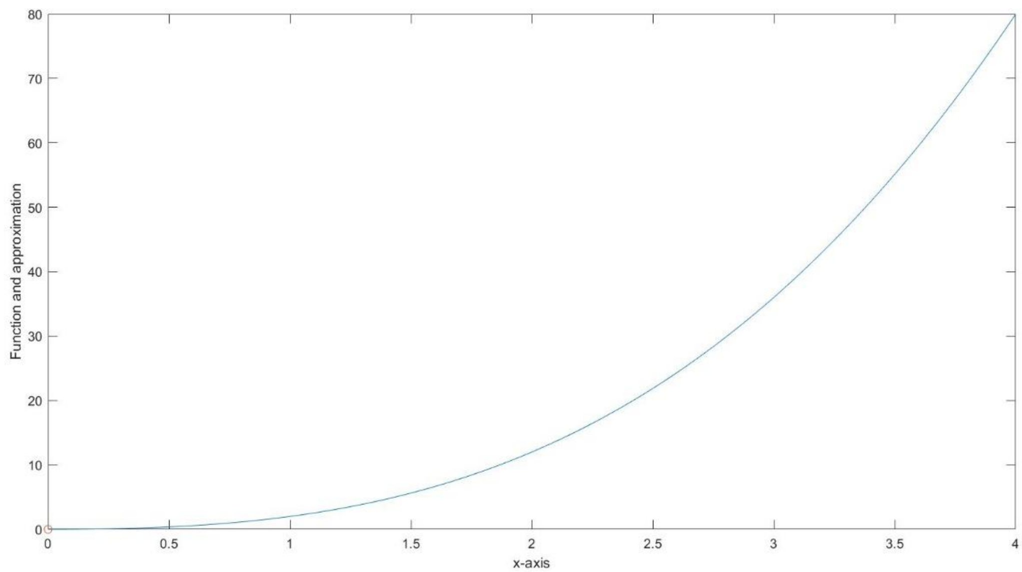
```
f = function_handle with value:
    @(x)(x^2+x^3)
Optimal x* = -0.0023
No. of iterations = 6
```

(c) NEWTON METHOD

```
f = @(x) (x^2 + x^3);
% x_(n+1) = x_(n)  -  f'(x_n)/f''(x_n)
f1 = @(x) 3*x^2  + 2*x;  % first derivative
f2 = @(x) 6*x + 2;   % second derivative
x = 0.1; % inntial guess
xx(1) = x;
err = 0.01; % tolerance
N= 100 ; % number of Iterations
for i = 1:N
    x =x - f1(x)/f2(x) ;
    j = i +1 ;
    xx(j) = x; ii = i;
    Err = abs(xx(j) - xx(j-1));
    if Err<err , break
    end
end
    FF = f(x);
    disp(['The Mininum Value at : ' num2str(x) ', is: ' num2str(FF) ', No of 
Iterations: ' num2str(ii)])
    xvalues = 0:0.01:4;
    F = xvalues.^2 + xvalues.^3 ;
    plot(xvalues,F);  % plot the given function
    hold on
    plot(x,FF,"o")
```

OUTPUT

```
The Mininum Value at: 5.5854e-08, is: 3.1197e-15, No of Iterations: 3
```



(d) PLOTTING

This is given above already for each methods.

(e) Testing Part a and b

```
% part a
fun = @(x) (x.^2 + x.^3);
x1 = 0;
x2 = 2;
options = optimset("Display","Iter");
x_min = fminbnd(fun,x1,x2,options)
```

OUTOUT

```
 Func-count     x          f(x)         Procedure
    1        0.763932      1.02942        initial
    2         1.23607      3.41641        golden
    3        0.472136     0.328157        golden
    4        0.271957    0.0940748        parabolic
    5        0.138921    0.0219803        parabolic
    6       0.0615426   0.00402058        parabolic
    7       0.0214162  0.000468475        parabolic
    8      0.00525533  2.77636e-05        parabolic
    9     0.000805899  6.49997e-07        parabolic
   10     0.000839233  7.04902e-07        parabolic
   11     0.000772566  5.97319e-07        parabolic
   12     0.000477472  2.28088e-07        golden
   13     0.000295094  8.71061e-08        golden
```

```
   14     0.000182378   3.32678e-08         golden
   15     0.000112716   1.27063e-08         golden
   16     6.96622e-05   4.85316e-09         golden
   17     3.63289e-05   1.31984e-09         golden

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
x_min = 3.6329e-05
```

```
% part b
x =[1  -0.5  2];
y = x.^2 + x.^3 % the function
y_inter = polyfit(x,y,2); % quadratic interpolation
f1 = @(x1) 2*x1.^2  + 0.125*x1 + 12; % The new quadratic function
x_min = fminbnd(f1,0,2)
```

OUTPUT

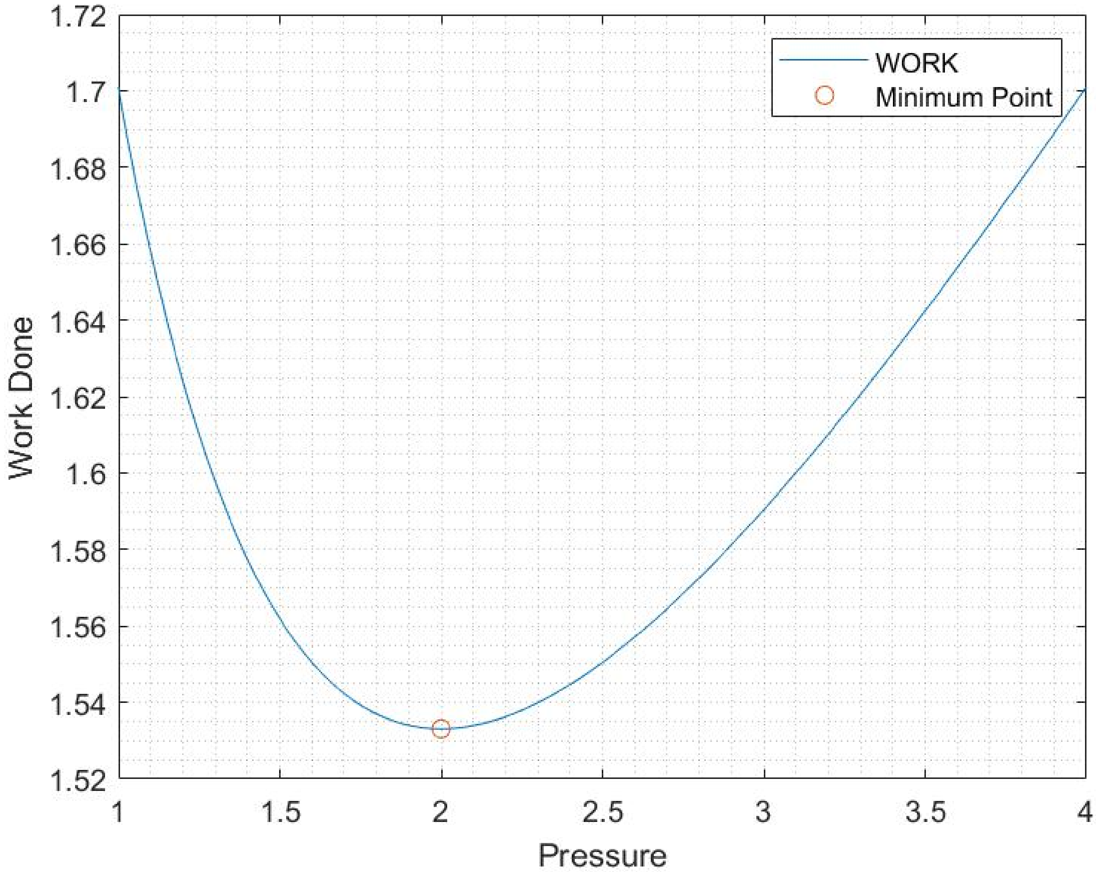
```
y = 1×3
    2.0000    0.1250   12.0000
x_min = 4.8379e-05
```

QUESTION 2

QUESTION a

```
p1 = 1;
p3 = 4;
gamma = 1.4 % gas
V1 = 1;
p = linspace(p1,p3);
f =  ((gamma*p1*V1)./(gamma-1))*(((p./p1)).^((gamma-1)./(gamma))  -2  + ...
     (p3./p).^((gamma-1)./(gamma)));
plot(p,f);
hold on
i=find(f(1,:)==min(f)); % find the point where W is minimum
j = p(1,34);% return the value of p in which W is minimum
plot(p(1,34),f(1,34),"o")  % mark the minimum point
xlabel("Pressure"),ylabel("Work Done");
grid minor
legend("WORK","Minimum Point")
hold off
```

OUTPUT



Question b

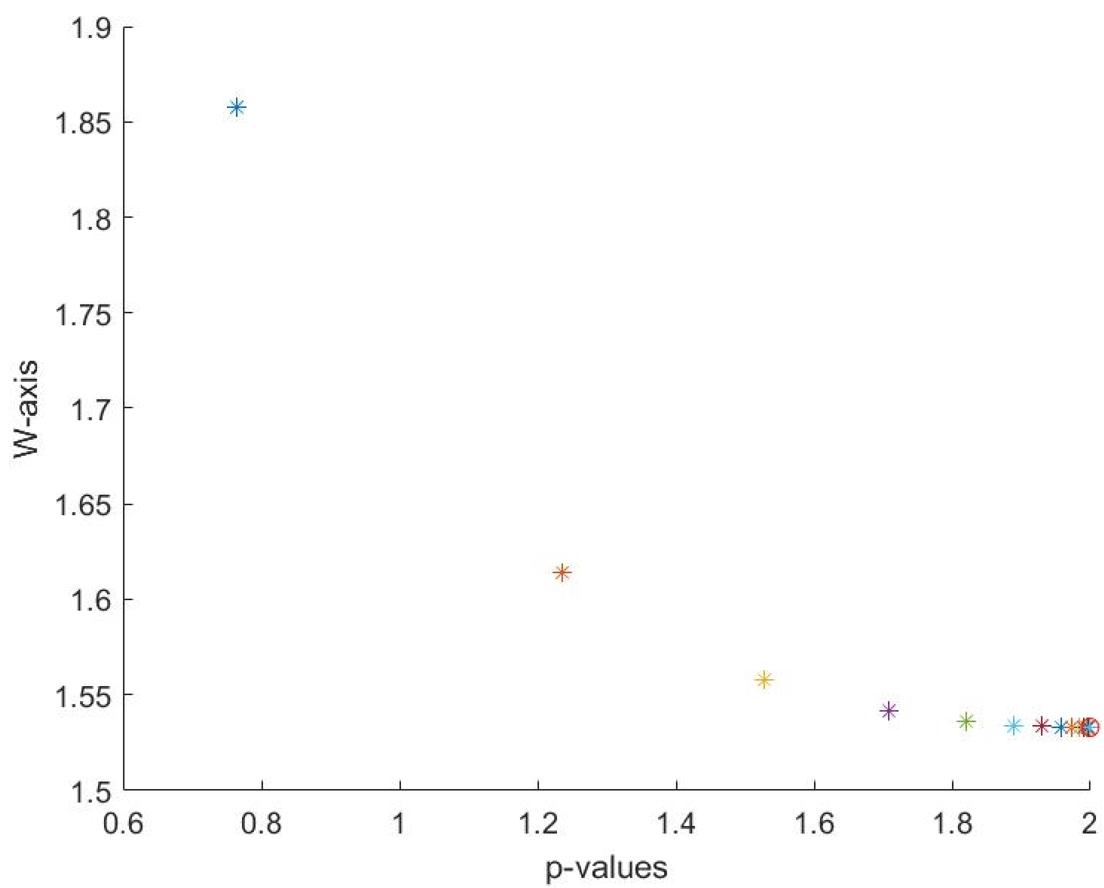
```
% Let x = p
figure; hold on;
a=0;                              % start of interval
b=2;                              % end of interval
epsilon=0.00001;                 % accuracy value
iter= 50;                         % maximum number of iterations
tau=double((sqrt(5)-1)/2);        % golden proportion coefficient, around 0.618
k=0;                              % number of iterations
f = @(x) ((gamma*p1*V1)./(gamma-1))*(((x./p1)).^((gamma-1)./(gamma))  -2  + ...
     (p3./x).^((gamma-1)./(gamma)));
x1=a+(1-tau)*(b-a);               % computing x values
x2=a+tau*(b-a);
```

```
f_x1=f(x1);                         % computing values in x points
f_x2=f(x2);


plot(x1,f_x1,'*')                  % plotting the function with x_min
plot(x2,f_x2,'*')

xlabel("p-values"), ylabel("W-axis");


while ((abs(b-a)>epsilon) && (k<iter))
    k=k+1;

    if(f_x1<f_x2)

        b=x2;

        x2=x1;

        x1=a+(1-tau)*(b-a);


        f_x1=f(x1);

        f_x2=f(x2);


        plot(x1,f_x1,'*');

    else

        a=x1;

        x1=x2;

        x2=a+tau*(b-a);


        f_x1=f(x1);

        f_x2=f(x2);


        plot(x2,f_x2,'*')

    end


    k=k+1;

end
```

```
% chooses minimum point
if(f_x1<f_x2)

    sprintf('p_min=% f', x1)

    sprintf('W(x_min)=% f ', f_x1)

    plot(x1,f_x1,'ro')

else

    sprintf('p_min=% f', x2)

    sprintf('W(p_min)=% f ', f_x2)

    plot(x2,f_x2,'ro')

end

disp("50 Iterations")
```

OUTPUT

```
gamma = 1.4000

ans = 'p_min= 1.999995'
ans = 'W(p_min)= 1.533096 '
```




```
50 Iterations
```

Question c

```
f = @(x) ((gamma*p1*V1)./(gamma-1))*(((x./p1)).^((gamma-1)./(gamma))  -2  + ...
     (p3./x).^((gamma-1)./(gamma)));

% The x-points values

x1 = 1;

x2 = -0.5;

x3 = 2;

tol = 10^-6;


% Invoking function
quadratic_interpolation(f, x1, x2, x3, tol)
```

I created a function below

```
function quadratic_interpolation(f, x1, x2, x3, tol)
fx1 = f(x1);

    fx2 = f(x2);

    fx3 = f(x3);

    if (fx1 > fx2 && fx2 < fx3)

        cond = true;

        iterations = 1;

        while(cond)

            syms a b c

            fx1 = f(x1);

            fx2 = f(x2);

            fx3 = f(x3);

            % Quadratic interpolation in the form of a second-order polynomial
y(x) = a + bx + cx^2

            % a, b, c is calculated from the following linear equations

            eq1 = a + x1*b + x1.^2 * c == fx1;

            eq2 = a + x2*b + x2.^2 * c == fx2;

            eq3 = a + x3*b + x3.^2 * c == fx3;

            sol = solve([eq1, eq2, eq3], [a, b, c]);

            aS = sol.a;
```

```
            bS = sol.b;
            cS = sol.c;

            xOpt = -(bS/(2*cS));
            parabolaOpt = aS + bS*xOpt + cS*(xOpt^2);
            fxOpt = f(xOpt);
            vars = [x1 x2 x3 xOpt];

            cond = (abs(fxOpt - parabolaOpt) > tol);
                if (cond)
                    iterations = iterations + 1;
                    % Calculating new points
                    vSorted = sort(vars);
                    vSIndice = find(vSorted==xOpt);
                    vSortedLeft = vSorted(:,1:(vSIndice-1));
                    vSortedRight = vSorted(:, (vSIndice+1:end));
                    x1 = max(vSortedLeft);
                    x2 = xOpt;
                    x3 = min(vSortedRight);
                end
        end
            fprintf('Optimal x* = %.4f \n', x2)
            fprintf('No. of iterations = %d \n', iterations)
    else
        disp('f(x1) > f(x2) && f(x2) < f(x3) not satisfied')
    end
end
```

OUTPUT

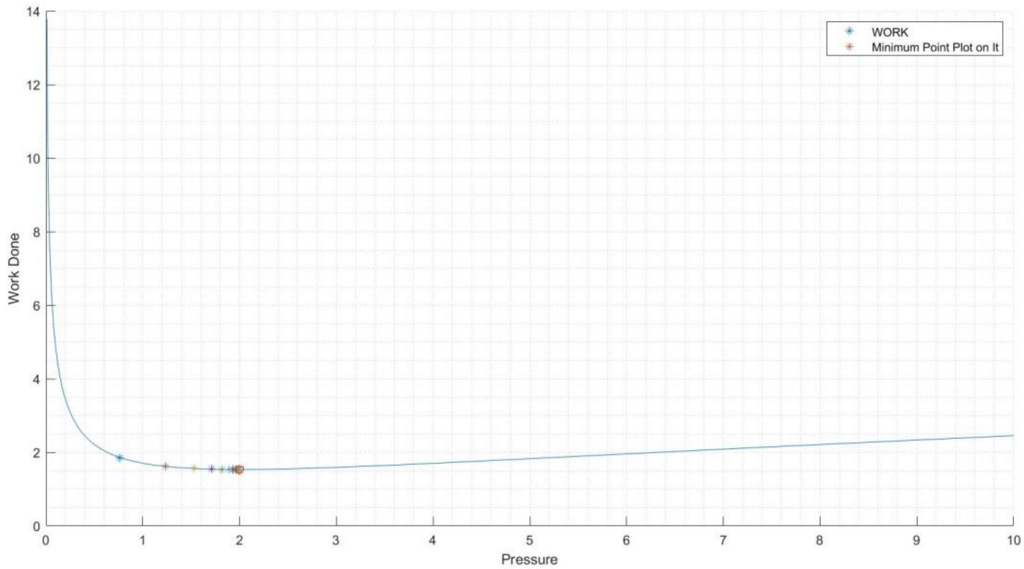
```
Optimal P* = 1.4837
No. of iterations = 2
```

Question d- Using An Appropriate Method

```
fun =@(x) ((gamma*p1*V1)./(gamma-1))*(((x./p1)).^((gamma-1)./(gamma))  -2  + ...
     (p3./x).^((gamma-1)./(gamma))) ;
x1 = 0;
x2 = 2;
options = optimset("Display","Iter");
p_min = fminbnd(fun,x1,x2,options)
% Plot the Optimal Solution on The Curve in part A
fplot(fun,[0,10]);
hold on
plot(p_min,f(p_min),"V")  % mark the minimum point
xlabel("Pressure"),ylabel("Work Done");
grid minor
legend("WORK","Minimum Point Plot on It")
hold off
```

OUTPUT

```
 Func-count     x          f(x)         Procedure
    1        0.763932      1.85774        initial
    2         1.23607      1.61387        golden
    3         1.52786      1.55836        golden
    4         1.60468      1.54999        parabolic
    5         1.75568      1.53901        golden
    6         1.90894      1.53385        parabolic
    7         1.96325      1.53322        parabolic
    8         1.99157       1.5331        parabolic
    9          1.9989       1.5331        parabolic
   10         1.99991       1.5331        parabolic
   11         1.99988       1.5331        parabolic
   12         1.99994       1.5331        parabolic

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of
1.000000e-04
p_min = 1.9999
```



WORK
Minimum Point Plot on It
Work Done
Pressure
0
2
4
6
8
10
12
14
0
1
2
3
4
5
6
7
8
9
10