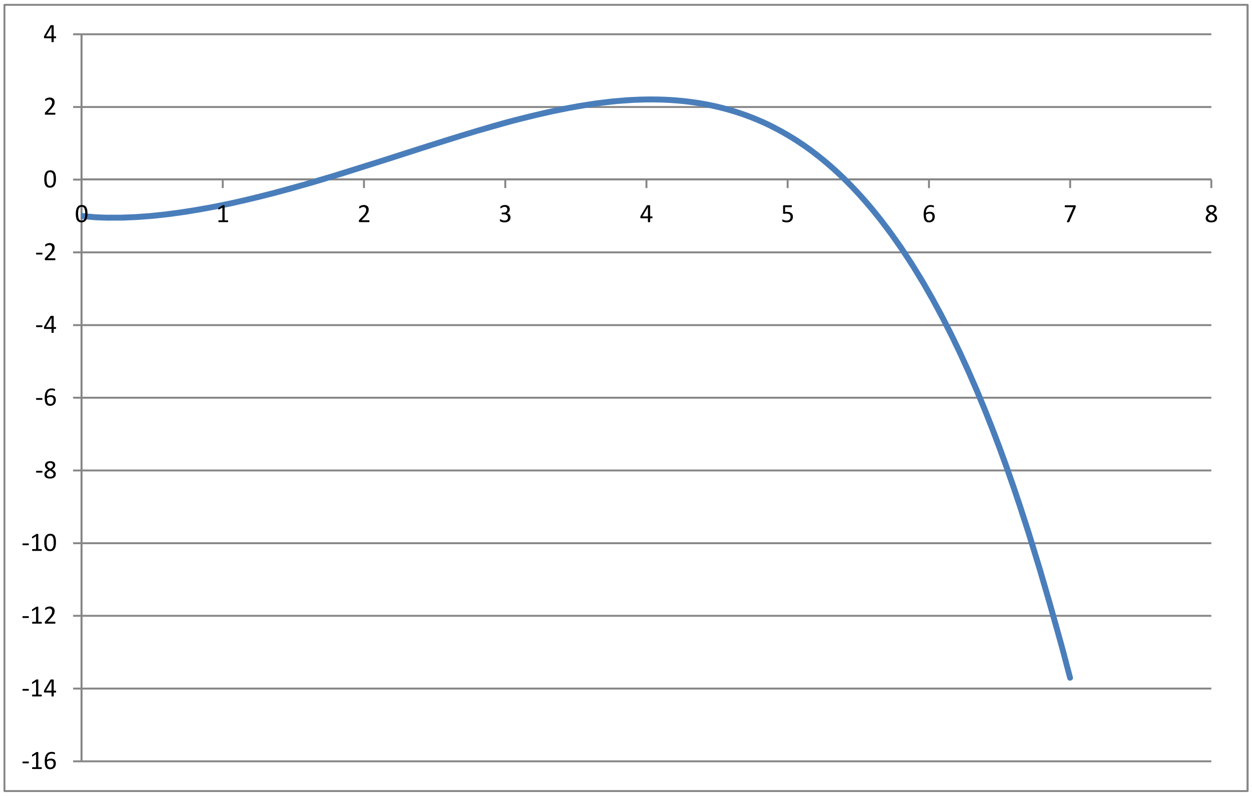
Assignment 8 Solution

Problem 1

Before we use bisection method to solve the problem, we can check the number of solutions and the intervals by looking at the graph. The graph shows that two solutions exist on the interval [1, 2] and [5, 6].



Matlab Code (Main Code for bisection search)

```
e =10^(-5);
xl = 1;
xr = 2;
xm = (xl + xr)/2;
while abs(xr - xl)> e
    if g(xm) == 0    % Stop if the xm makes the function value = 0
        break;
    else
        if g(xm) >0   % Choose which side to update
            xr = xm;
        else
            xl = xm;
        end
    end
    xm = ( xl + xr)/2;   % Update xm according to the new interval
end
result = xm
```

Matlab code (Function File)

```
function result=g(a)
result = a^(1.7)-1.7^a;
```

result =

1.70000

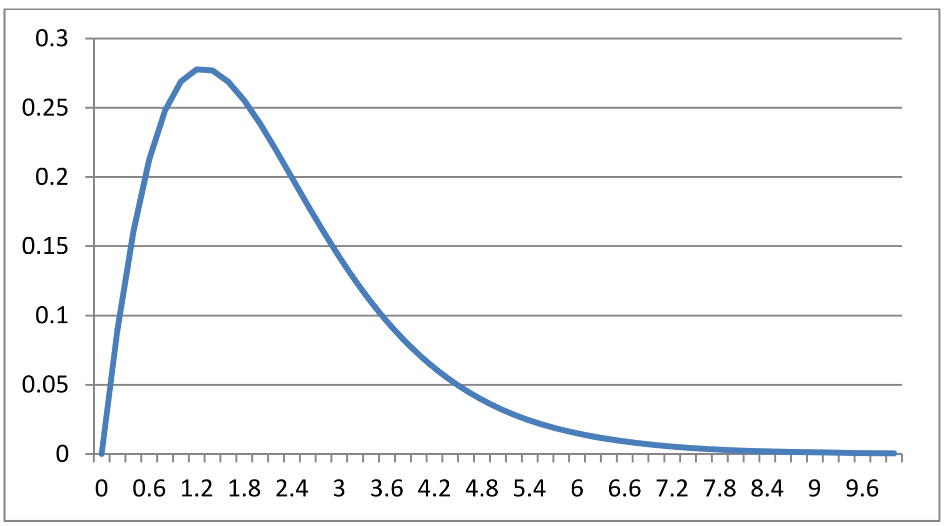
Apply the same code for xl = 5 and xr = 6, but here we need to switch the updating policy on both side, then we get

result =

5.40687

Problem 2

Since it is a unimodal function on [0,10], we can use golden search method.



Matlab code (Main Code Golden Section Search)

```
e = 10^(-5);
xl = 0;
xr = 10;
phi = (3-sqrt(5))/2;
while xr – xl > e
    xl1 = phi * xr + (1-phi) * xl;      % find a left point in the section
using the golden ratio.
    xr1 = (1-phi) * xr + phi * xl;
    if f(xl1) > f(xr1)   % If the function value is greater toward the left,
        xr = xr1;      % then discard the right most interval.
```

```
    else
        xl = xl1;
    end
end
result = (xl + xr)/2;
```

Matlab code (Function File)

```
function result = f(a)
result = a * exp(-a)/(1 + exp(-a));
```

golden

ans =

   1.2785

Question 3

-Backtracking search method:

Matlab code (Function File)

```
function y = f(x)
y = exp(1-x(1)-x(2))+ exp(x(1)+x(2)-1)+x(1)^2+x(1)*x(2)+x(2)^2+2*x(1)-3*x(2);
```

Matlab code (Gradient Vector)

```
function y = gradient(x)

y1 = -exp(1-x(1)-x(2)) + exp(x(1) + x(2)-1) + 2*x(1) + x(2) + 2;
y2 = -exp(1-x(1)-x(2)) + exp(x(1) + x(2)-1) + x(1) + 2*x(2) - 3;
y = [y1; y2];
```

Matlab code (Main Code for backtracking search method)

```
x = [0; 0];
epsilon = 10^-5;
iter = 0;
alpha = 0.5;
beta = 0.5;
while norm(gradient(x)) > epsilon
    % Start doing backtracking search
    t = 1;
    xtemp = x - t * gradient(x);
    while f(xtemp) >= f(x) - alpha * t * norm(gradient(x))^2    % if the
function value increased, we try another step size.
        t = t * beta; % update the step size
        xtemp = x - t * gradient(x);
```

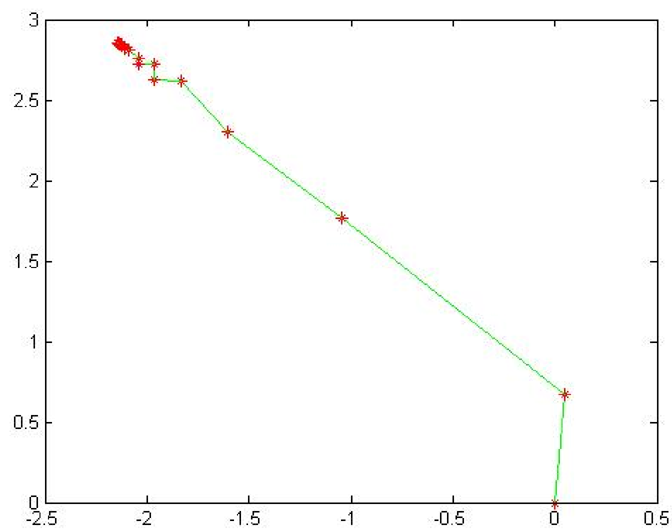
```
    end
    % Make some plots
    plot(x(1), x(2), '*r');
    hold on;
    plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-g');
    hold on;
    % Output the solution in each step
    iter = iter + 1;
    x = xtemp;
end
```

gradient_descent

ans =

-2.1418

2.8582



-Exact Line search method using gradient descent method:

Matlab code (Main) (other two codes remain same)

```
x = [0; 0];
epsilon = 10^-5;
iter = 0;
e = 10^(-5);
phi = (3-sqrt(5))/2;      % Here, we use Golden Section Search to find the
minimizer

while norm(gradient(x)) > epsilon
    % Start exact search
    alphal = 0;
```

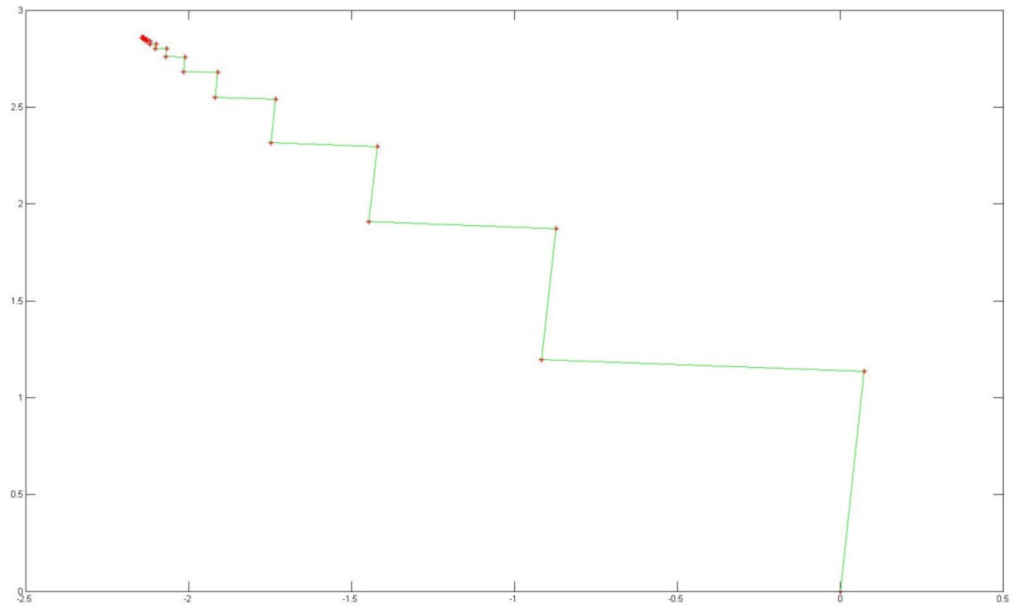
```
    alphar = 10;
    while alphar – alphal > e
        alphal1 = phi * alphar + (1-phi) * alphal;    % Use Golden Section
Search. First calculate two points
        alphar1 = (1-phi) * alphar + phi * alphal;
        xtemp1 = x - alphal1 * gradient(x);
        xtemp2 = x - alphar1 * gradient(x);
        if f(xtemp1) <= f(xtemp2)
            alphar = alphar1;              % Update the points
        else
            alphal = alphal1;
        end
        alpha=(alphal + alphar)/2;
    end
    xtemp = x – alpha * gradient(x);
    % Make plots
    plot(x(1), x(2), '*r');
    hold on;
    plot([x(1), xtemp(1)], [x(2), xtemp(2)], '-g');
    hold on;
    % Output the solution in each step
    iter = iter + 1;
    x = xtemp;
end
```

gradient_exact

ans =

   -2.1418

    2.8582


3
2.5
2
1.5
1
0.5
0
-2.5
-2
-1.5
-1
-0.5
0
0.5