6/5/2020

Sungkyunkwan University

School of Information and communication Engineering

Course title: Optimization Methods

Course code: ECE5920(41)

Term Project

Muleta, Kalkidan Deme
ID: 2017712139

Optimization Methods Term Project

Notes:

The first problem is implemented on MATLAB, all the codes are attached here and are also available on GitHub with this link. `GoldenSectionFinal.m` and `NewtonMethodFinal.m` are the main functions while `f.m`, `fPrime.m` and `f2Prime.m` are supplementary MATLAB functions to implement the given *function*, its *first* and *second derivatives*.

The second problem can be found on the last page here. It contains a clickable google link to view the plot.

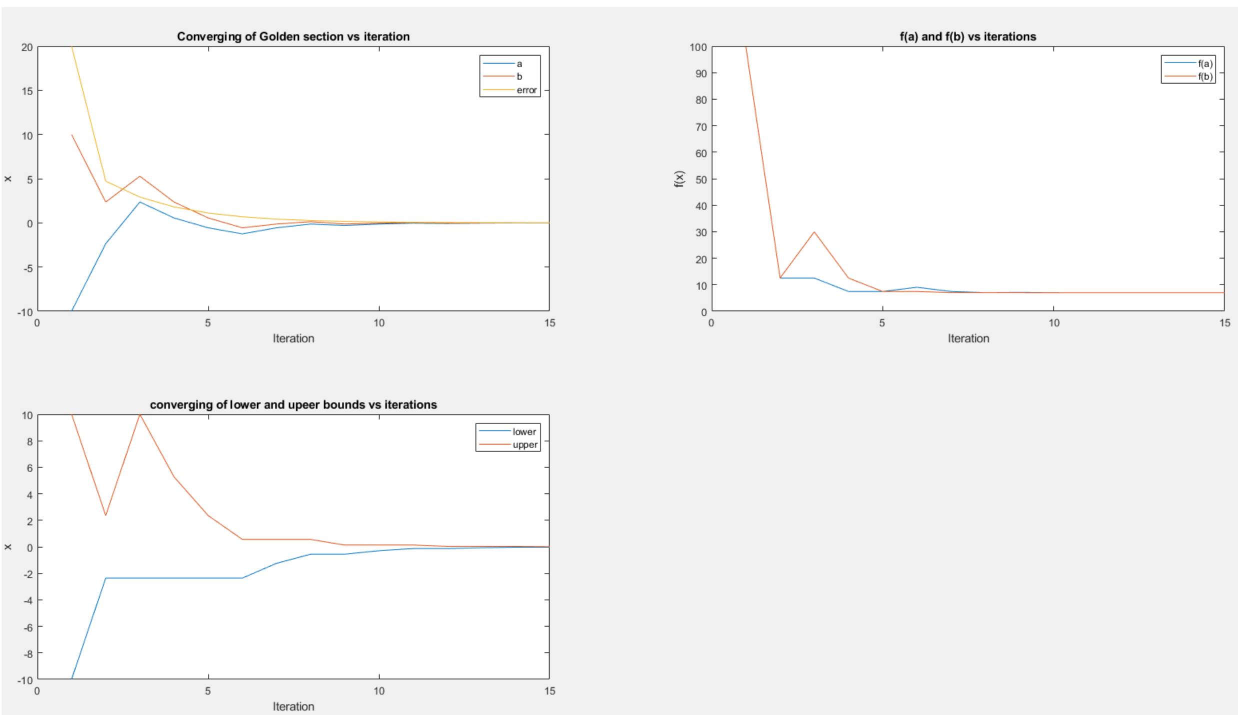
```matlab
%{
    Description: evaluates the f(x) for a given x
          Input: x
         Return: f(x)
%}
function [f_x] = f(x)
    f_x = x^2 + x*sin(x) + cos(x) + 6;
end


%{
    Description: evaluates the first derivative of f(x) for a given x
          Input: x
         Return: f'(x)
%}

function f_prime = fPrime(x)
    f_prime =  2*x + x*cos(x);
end


%{
    Description: evaluates the second order derivative of f(x) for a given x
          Input: x
         Return: f"(x)
%}

function f_2Prime = f2Prime(x)
    f_2Prime = cos(x) - x*sin(x) + 2;
end
```

```matlab
%{
          Title : Golden Section Method
    Description : Implementation of Golden Section Method for minimizing the
                  function f(x) = x^2 + x*sin(x) + cos(x) + 6 over the
                  interval [-10 10] with error less than 0.1%.
  Obj. function : f(x) = x^2 + x*sin(x) + cos(x) + 6
       interval : [-10 10]

        Note: f(x) is implemented in a separate matlab function script and
              is simply invoked here

     Written by : Muleta Kalkidan Deme, ID : 2017712139,
                   Sungkyunkwan University
           Date : Jun 1, 2020
%}

clear
clc
%interval
lower = -10;                 % Lower boundary(a0)
upper = 10;                  % Upper boundary(b0)
error = upper - lower;       % Initialize the error equal to full range
Interval = [lower upper];    % Initial interval

%initilize
p = 0.3820;                  % The Golden Ratio
                                     % First iteration
a  = lower + p*(upper-lower);        % a1
b = lower + (1-p)*(upper-lower);     % b1
f_a = f(a);                          % f(a1)
f_b = f(b);                          % f(b1)

% keep iteration records to be used in tables later
As(1,1) = lower;
As(2,1) = a;
Bs(1,1) = upper;
Bs(2,1) = b;
Es(1,1) = error;
Es(2,1) = b-a;
f_As(1,1) = f(lower);
f_As(2,1) = f_a;
f_Bs(1,1) = f(upper);
f_Bs(2,1) = f_b;
Intervals(1,1:2) = Interval;
Intervals(2,1:2) = [a b];

while error > 0.01           % until acceptable error
   if f_a < f_b              % if f(ai) < f(bi), on each iteration
      upper = b;             % new upper boundary
      b = a;                 % next value of b
      f_b = f_a;
      a = lower + p*(upper-lower);% next value of a,
      f_a = f(a);

   else                      % if f(ai) >= f(bi)
       lower = a;            % new lower boundary
       a = b;                % next value of a, a(i+1) = b(i)
       f_a = f_b;
       b = lower + (1-p)*(upper-lower); % next value of b
       f_b = f(b);
   end
   error = b-a;     % error of the current iteration
   Interval = [lower upper]; % updated interval

   % table values
   As(end+1,1) = a;
   Bs(end+1,1) = b;
   Es(end+1,1) = error;
   f_As(end+1,1) = f_a;
   f_Bs(end+1,1) = f_b;
   Intervals(end+1,1:2) = Interval;
end
table(As,Bs,f_As,f_Bs,Es,Intervals)
```

```
% plots
figure('Name','Golden Section Method','NumberTitle','off');
subplot(2,2,1)
plot(As)
title('Converging of Golden section vs iteration')
hold on
plot(Bs)
plot(Es)
hold off
xlabel('Iteration')
ylabel('x')
legend('a','b','error')

subplot(2,2,2)
plot(f_As)
title('f(a) and f(b) vs iterations')
hold on
plot(f_Bs)
xlabel('Iteration')
ylabel('f(x)')
legend('f(a)','f(b)')

subplot(2,2,3)
plot(Intervals)
title('converging of lower and upeer bounds vs iterations')
xlabel('Iteration')
ylabel('x')
legend('lower','upper')

```

Results of Golden section method

As	Bs	f_As	f_Bs	Es	Intervals	
-10	10	99.721	99.721	20	-10	10
-2.36	2.36	12.522	12.522	4.72	-2.36	2.36
2.36	5.2785	12.522	29.944	2.9185	-2.36	10
0.5579	2.36	7.455	12.522	1.8021	-2.36	5.2785
-0.55696	0.5579	7.4535	7.455	1.1149	-2.36	2.36
-1.2454	-0.55696	9.0506	7.4535	0.6884	-2.36	0.5579
-0.55696	-0.13095	7.4535	7.0257	0.42601	-1.2454	0.5579
-0.13095	0.13202	7.0257	7.0261	0.26297	-0.55696	0.5579
-0.29377	-0.13095	7.1285	7.0257	0.16282	-0.55696	0.13202
-0.13095	-0.030629	7.0257	7.0014	0.10032	-0.29377	0.13202
-0.030629	0.031569	7.0014	7.0015	0.062198	-0.13095	0.13202
-0.068866	-0.030629	7.0071	7.0014	0.038237	-0.13095	0.031569
-0.030629	-0.0067973	7.0014	7.0001	0.023832	-0.068866	0.031569
-0.0067973	0.007809	7.0001	7.0001	0.014606	-0.030629	0.031569
-0.015946	-0.0067973	7.0004	7.0001	0.0091485	-0.030629	0.007809



```matlab
%{
          Title : Newton Method
    Description : Implementation of Newton Method for minimizing the
                  function f(x) = x^2 + x*sin(x) + cos(x) + 6 with initial
                  points being 1) x_i = -10 or 2) x_i = 10,  with error
                  less than 0.1%.
  Obj. function : f(x) = x^2 + x*sin(x) + cos(x) + 6
  initial point : 1) x_i = -10
                  2) x_i = 10

        Note: f(x, f'(x) and f"(x) are implemented in three separate matlab
                function script, and are simply invoked in this script

     Written by : Muleta Kalkidan Deme, ID : 2017712139,
                   Sungkyunkwan University
           Date : Jun 1, 2020
%}

clear
clc

%Initialize
x_start = [-10 10];                  % two starting points

for i=1:length(x_start)
    %clear
    error = 2;                       % reset error for new starting point
    x = x_start(i);                  % current starting point
    iter = 0;
    Xs(1,i) = x;                     % records the x, used for table & plots
    f_Xs(1,i) = f(x);                % records the f(x)
    while error > 0.01
        x_new = x - fPrime(x)/f2Prime(x);   % new x
        error  = abs(x_new - x);            % error
        x = x_new;
        if i  == 1                          % for the first starting point
            Xs(end+1,i) = x_new;
            f_Xs(end+1,i) = f(x_new);
        else
            Xs(iter+1,i) = x_new;
            f_Xs(iter+1,i) = f(x_new);
        end
        iter = iter +1;
    end
    % table
    table(Xs(:,i),f_Xs(:,i))

end
% plots
figure('Name','Newton Method ','NumberTitle','off');
hold on

subplot(2,2,1)
plot(Xs(:,1))
title('Newton Method, x_start = -10 ')
ylabel('x')
xlabel('Iteration')
legend('x')

subplot(2,2,2)
plot(f_Xs(:,1))
title('Newton Method, x_start = -10')
ylabel('f(x)')
xlabel('Iteration')
legend('f(x)')

subplot(2,2,3)
plot(Xs(:,2))
title('Newton Method, x_start = 10 ')
ylabel('x')
xlabel('Iteration')
legend('x')

```

```
subplot(2,2,4)
plot(f_Xs(:,2))
title('Newton Method, x_start = 10 ')
ylabel('f(x)')
xlabel('Iteration')
legend('f(x)')
```

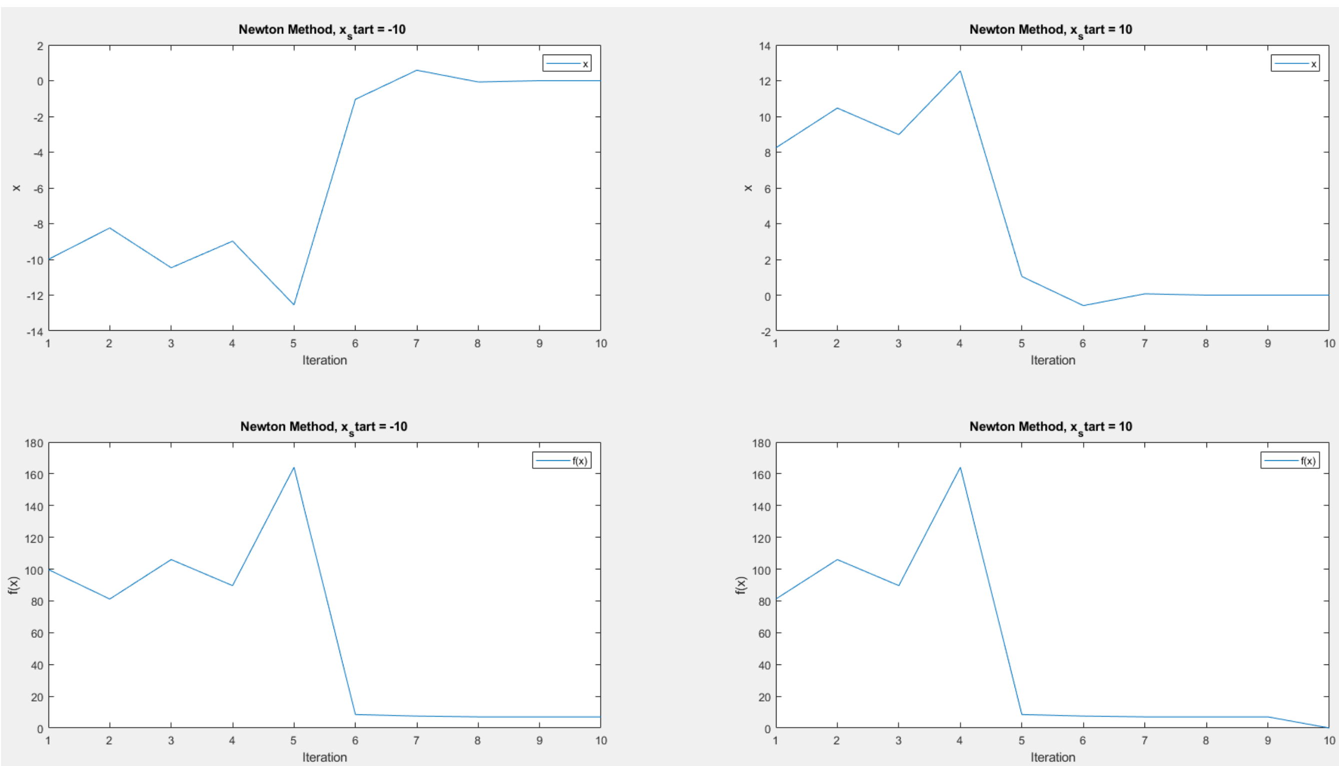
Results for Newton's method

a) x_start = -10

Var1	Var2
-10	99.721
-8.2413	81.172
-10.466	106.01
-8.9803	89.604
-12.545	164.1
-1.0421	8.4902
0.58437	7.4979
-0.075006	7.0084
0.00014092	7
-9.3284e-13	7

b) x_start = 10

Var1	Var2
8.2413	81.172
10.466	106.01
8.9803	89.604
12.545	164.1
1.0421	8.4902
-0.58437	7.4979
0.075006	7.0084
-0.00014092	7
9.3284e-13	7
0	0



Problem #2

Let's first define the Lagrange function

$$\mathcal{L} = f(x) + \lambda^T h(x)$$
$$\mathcal{L} = x_1 x_2 + \lambda(x_1 + x_2 - 1)$$

The partial derivation of the Lagrange function in terms of **x** and λ

$$D_{x_1}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial x_1} = x_2 + \lambda = 0$$
$$D_{x_2}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial x_2} = x_1 + \lambda = 0$$
$$D_{\lambda}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial \lambda} = x_1 + x_2 - 1 = 0$$

By expressing the first and second equations in terms of x_1 and x_2 respectively, and inserting it into the third equation, we have

$$-\lambda - \lambda - 1 = 0$$
$$\lambda = -0.5.$$

Thus, $x_1 = x_2 = 0.5$. i.e. $x = [\,0.5 \;\; 0.5]^T$

We can plot the function on MATLAB or google and we can see the curve continues to go up and down, and the point where the gradient will be zero at is at $x_1 = x_2 = 0.5$ for $\lambda = -0.5$, but this is neither the maxima nor the minima of the function.