2021

Homework 4

AE_5830 DR. HOSDER

MATT PAHAYO

AE/ME 5830 Spring 2021, Homework IV, Due Wednesday March 31 by midnight

1. Develop a computer routine to minimize a one-dimensional function $F(x)$ in positive x domain. Your routine should include four parts:

   (a) Finding the bounds on the minimum of the function assuming that the function has a negative slope at x=0.0.

   (b) Reduction of the original interval found in Part (a) using the golden section algorithm.

   (c) Cubic polynomial fit to the points obtained at the last iteration of the golden section algorithm.

   (d) Determining the location of the minimum (x_{min}) and the corresponding value of the objective function (f_{min}).

   Use your routine to find the minimum of the following function:

$$F(x) = x^4 - x^3 - Sin^2x + Cos^2x + 2 \tag{1}$$

   Solve the problem for n = 2, n = 5, n = 10, and n = 15 where n is the number of iterations for the golden section search. For each case, report x_{min} and f_{min}. (Hint: Use a = 0.0 and b = 0.1 for the starting values of the bounds in part (a)).

2. The drag (D) of a wide-body passenger aircraft can be estimated by

$$D = 6.62725\sigma V^2 + \frac{1.31493 \times 10^{-4}}{\sigma}\left(\frac{W}{V}\right)^2 \tag{2}$$

   where σ = ratio of air density between the flight altitude and sea level, W = weight of the aircraft in Newtons, V = velocity of the aircraft in m/s, and D is obtained in Newtons. In the above equation, the first term corresponds to the drag due to friction and the second term represents the drag due to lift. At a given altitude and aircraft weight, there will be an optimum value of the velocity which will minimize the total drag (e.g., maximize L/D value). Using the optimization routine you have written, determine the minimum drag and the corresponding velocity for this aircraft at cruise weight and altitude (W = 3.7278×10^6 N and σ = 0.31). Use 100 m/s ≤ V ≤400 m/s for the initial interval in the golden section search and use a relative convergence criteria of $\epsilon = 10^{-3}$ for interval reduction.

Results

Q1:

Table 1 Solution to Question 1

iterations	*xmin*	*fmin*	a	c	d	b
0	0.0172	2.9994	0	0.0382	0.0618	0.1
2	0.0106	2.9998	0	0.0236	0.0382	0.0618
5	0.0025	3	0	0.0056	0.009	0.0146
10	2.26E-04	3	0	5.03E-05	8.13E-04	1.30E-03
15	2.04E-05	3	0	4.53E-05	7.33E-05	1.18E-04

Q2:

Turn the function negative to make interval work

Number of iterations = 14.35; round up

Velocity = 231.579m/s

Drag = 2.009e5 N, note the function was flipped beforehand

Table 2 Velocity Interval in m/s

a	c	d	b
231.3082	231.4442	231.5282	231.6641

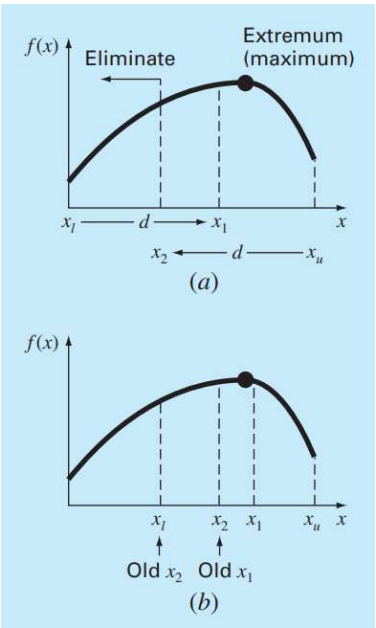

Fig.1 GS Method

Methodology

A. Development of Golden section Algorithm

A. First find ‘d’ with equation 1.
B. Get the number of iterations from equation 2.
C. Get the new ‘d’ value with equation 3.
D. Compare the lower and the upper bounds of the function.
E. Adjust values accordingly. See figure 2.

$$d = R(x_b - x_{a)} \qquad (1)$$

$$N = \frac{\log(\varepsilon)}{\log(R)} \qquad (2)$$

$$d = Rd \qquad (3)$$

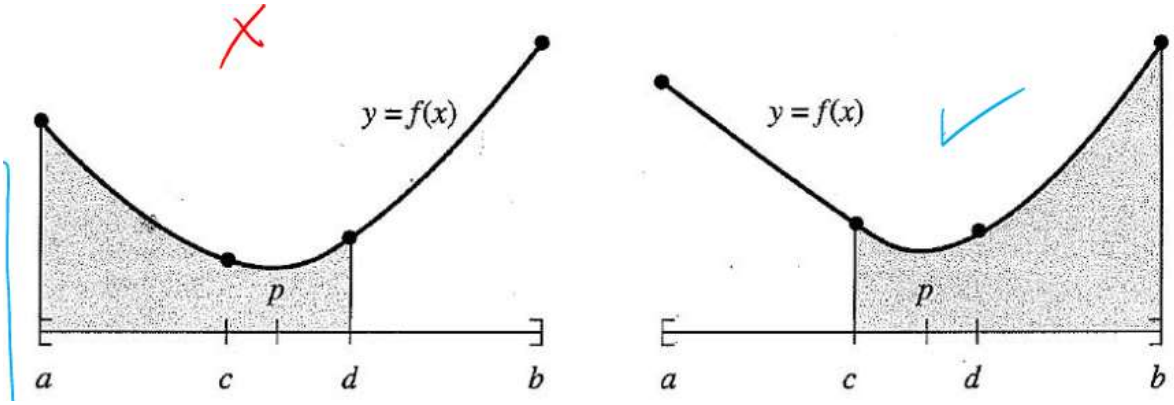

Fig. 2 Graphic for Choosing New Bounds

F. If f(c) is less than f(d) make the left graphic the new bounds, that is ‘d’ becomes b and ‘c’ becomes ‘d’ and recalculate for the new value of ‘d.’
G. Repeat C-F until number of iterations is met

APP. A

```
classdef onedOpt
    % onedOpt is a class of 1-d optimization functions

    methods (Static)
        function [xlow,x2,x1,xhigh] = Gold(xlow,xhigh,n,es,f)
            % Gold is the Golden section algorithm for 1-d opt.
            % if using a set convergence target set es to the according
            % value and set iter to any value else set es to zero for a
targeted amount of iterations
            R = (sqrt(5)-1)/2;
            if es>0
               n = log10(es)/log10(R)
            end
            iter = 1;
            d = R*(xhigh-xlow);
            x1 = xlow + d;
            x2 = xhigh - d;
            f1 = double(f(x1));
            f2 = double(f(x2));
            if f1>f2
                xopt = x1;
                fx = f1;
            else
                xopt = x2;
                fx = f2;
            end

            while iter<n
                d = R*d;
                if f1>f2
                    xlow = x2;
                    x2 = x1;
                    x1 = xlow + d;
                    f2 = f1;
                    f1 = double(f(x1));
                else
                    xhigh = x1;
                    x1 = x2;
                    x2 = xhigh - d;
                    f1 = f2;
                    f2 = double(f(x2));
                end
                if f1>f2
                    fx = f1;
                else
                    fx = f2;
                end
                iter = iter + 1;
            end
        end


        function Newton = Newton()
            % Newton is the Newton Method for 1-d opt.
```

```
        end
    end
end
```

```
function [x,y] = cubicFit(xlow,x2,x1,xhigh,f)
%cubicFit fits a cubic function into to the specified points
%   takes values from Gold
q1 = x1^3*(x2-xlow)-x2^3*(x1-xlow)+xlow^3*(x1-x2);
q2 = xhigh^3*(x2-xlow)-x2^3*(xhigh-xlow)+xlow^3*(xhigh-x2);
q3 = (x1-x2)*(x2-xlow)*(x1-xlow);
q4 = (xhigh-x2)*(x2-xlow)*(xhigh-xlow);
q5 = double(f(x1))*(x2-x1)-double(f(x2))*(x1-xlow)+double(f(x1-x2));
q6 = double(f(xhigh))*(x2-x1)-double(f(x2))*(xhigh-xlow)+double(f(xhigh-x2));

a3 = (q3*q6-q4*q5)/(q2*q3-q1*q4);
a2 = (q5-a3*q1)/q3;
a1 = (double(f(x2)-f(xlow)))/(x2-xlow)-a3*(x2^3-xlow^3)/(x2-xlow)-...
    a2*(xlow+x2);
del = a2^2-3*a1*a3;

x(1) = double(-a2+sqrt(del))/3/a3;
x(2) = double(-a2-sqrt(del))/3/a3;
y(1) = double(f(x(1)));
y(2) = double(f(x(2)));
end
```

```
% matthew Pahayo
% main.m
clc
clear all
close all

format longg
syms x V

%=========================================================================%
                        % q1
%=========================================================================%
% f = symfun(2*sin(x)-x^2/10,x)
f = symfun(x^4-x^3-(sin(x))^2+(cos(x))^2+2,x);
xlow = 0;
xhigh = .1;
iter = 10;
[xlow,x2,x1,xhigh] = onedOpt.Gold(xlow,xhigh,iter,0,f);
[x,y] = cubicFit(xlow,x2,x1,xhigh,f);

%=========================================================================%
                        % q2
%=========================================================================%
```

```
W = 3.7278*10^6; %[N]
sigma = .31;
D = symfun(-231.579(6.62725*sigma*V^2+1.31493*10^-4/sigma*(W/V)^2),V)
Vlow = 100;
Vhigh = 400;
es = 10^-3;
[Vlow,V2,V1,Vhigh] = onedOpt.Gold(Vlow,Vhigh,0,es,D);
[V,D] = cubicFit(Vlow,V2,V1,Vhigh,D);
```