SNEHA GANESH

ME 8710 ENGINEERING OPTIMIZATION

PROBLEM STATEMENT

Test the methods on the following function:

- Interval halving method. (0 order)
- Golden section method. (0 order)
- Bisection (1st order)
- Powell's method (sequential Quadratic)
- Cubic search

Function: $f(x) = 10 + (x - 4)^2 + 6 * EXP^{(-x/4 + 2)} - 40*COS(0.1*x*PI)$

Set a termination criterion (ε <= 0.00001), input the starting values as x_1 = -5 (for bracketing with a step size of 2) and x_2 = 20 and allow the user to modify the values.

MATLAB CODE

```
%UNCONSTRAINED SINGLE VARIABLE ONE DIMENSIONAL MINIMIZATION/OPTIMIZATION
METHODS
clc;
true=1;

while true
syms x x1 x2 xm;
m=0;
phi=0.6180334;
fprintf('UNCONSTRAINED SINGLE VARIABLE ONE DIMENSIONAL
MINIMIZATION/OPTIMIZATION METHODS \n 1.INTERVAL HALVING METHOD \n 2.GOLDEN
SEARCH METHOD')
fprintf('\n 3.BISECTION METHOD \n 4.POWELLS METHOD \n 5.CUBIC SEARCH METHOD
\n')
c = input('\nChoose Method: ');
f = input('\nEnter Function: ');
Tolerance = input('\nEnter tolerance value: ');
```

(INTERVAL HALVING METHOD)

```
%INTERVAL HALVING METHOD
if(c==1)
 a = input('\nEnter lower interval value: ');
 b = input('\nEnter upper interval value: ');
 L = b-a;
while abs(L)>=Tolerance
```

```
    m=m+1;
    xm = (a+b)/2;
    fxm = subs(f,xm);
    x1 = a+L/4;
    x2 = b-L/4;
    fx1 = subs(f,x1);
    fx2 = subs(f,x2);
    if fx1<fxm
        b=xm;
        xm = x1;
    elseif fx2<fxm
        a=xm;
        xm = x2;
    elseif fx2>=fxm
        a=x1;
        b=x2;
    end
    L = b-a;
end
m
vpa(xm)
vpa(fxm)
```

(GOLDEN SEARCH METHOD)

```
%GOLDEN SEARCH METHOD
elseif(c==2)

 a = input('\nEnter lower interval value: ');
 b = input('\nEnter upper interval value: ');
 L = abs(a-b);
 x1 = b-phi*(b-a);
 x2 = a+phi*(b-a);
 fx1= subs(f,x1);
 fx2= subs(f,x2);
 while L>Tolerance
   m=m+1;
     if(fx1>fx2)
          a=x1;
          x1=x2;
          fx1= subs(f,x1);
          x2=a+phi*(b-a);
          fx2= subs(f,x2);
      else
          b = x2;
          x2 = x1;
          fx2= subs(f,x2);
          x1 = b-phi*(b-a);
          fx1= subs(f,x1);
      end
  L=abs(b-a);
 end
 m
 vpa(x1)
 vpa(fx1)
```

(BISECTION METHOD)

```
%BISECTION METHOD
elseif(c==3)
 a = input('\nEnter lower interval value: ');
 b = input('\nEnter upper interval value: ');
 f_dot=diff(f,x)
 if(subs(f_dot,a) < 0 && subs(f_dot,b)>0)
   x1=a;
   x2=b;
   xm=(x1+x2)/2;
   xm_dot=subs(f_dot,xm);
while abs(xm_dot) > Tolerance
   if(xm_dot<0)
       x1=xm;
       xm=(x1+x2)/2;
       xm_dot=subs(f_dot,xm);
   elseif(xm_dot>0)
       x2=xm;
       xm=(x1+x2)/2;
       xm_dot=subs(f_dot,xm);
   end
   m=m+1;
end
 else
  fprintf('\nNot a valid bracket value')
 end
m
vpa(xm)
vpa(subs(f,xm))
```

(POWELL'S METHOD (SEQUENTIAL QUADRATIC))

```
%POWELL'S METHOD (SEQUENTIAL QUADRATIC)
elseif(c==4)
   syms x0;
   k=1;
   j=1;
   x0 = input('\nEnter start value: ');
   Step_Size = input('\nEnter step value: ');
   E_2= input('\nEnter termination parameter E2: ');
   x1=x0+Step_Size;
   fx0=subs(f,x0);
   fx1=subs(f,x1);
   if(fx0>fx1)
   x2= x0 + (2*Step_Size);
   else
   x2= x0 - (Step_Size);
   end
   X = [x0,x1,x2];
   T =sort(X);
   x0 = T(1,1);
   x1 = T(1,2);
   x2 = T(1,3);
   while(k >Tolerance && j>E_2)
```

```
        fx0=subs(f,x0);
        fx1=subs(f,x1);
        fx2=subs(f,x2);
        fmin = min([fx0,fx1,fx2]);
        if fmin==(fx0)
           xmin = x0;
        elseif fmin==(fx1)
           xmin = x1;
        elseif fmin==(fx2)
           xmin = x2;
        end
        a0=fx0;
        a1=(fx1-fx0)/(x1-x0);
        a2=(((fx2-fx0)/(x2-x0))-((fx1-fx0)/(x1-x0)))/(x2-x1);
        xm = (x1+x0)/2 - a1/(2*a2);
        fxm=subs(f,xm);
        k = abs((fmin - fxm)/fxm);
        j = abs(xmin - xm);
        best_point = min([xmin,xm]);
        if (best_point == x0 || best_point==x1 || best_point == x2)
        best_point = max([xmin,xm]);
        end
        if (best_point>x0 && best_point<x1)
            x2 = x1;
            x1 = best_point;
        elseif(best_point>x1)
            x0 = x1;
            x1 = best_point;
        end
        Y = [x0, x1, x2];
        S =sort(Y);
        x0 = S(1,1);
        x1 = S(1,2);
        x2 = S(1,3);
        m=m+1
    end
m
vpa(xm)
vpa(fxm)
```

(CUBIC SEARCH METHOD)

```
%CUBIC SEARCH METHOD
elseif(c==5)
  k=0;
  fxbardot = 1;
  fdot_x0 = 1;
  fdot_xk1 = 1;
  x0 = input('\nEnter start value: ');
  Step_Size = input('\nEnter step value: ');
  E_2= input('\nEnter termination parameter E2: ');
  while fdot_x0*fdot_xk1>=0
    fx0 = subs(f,x0);
    fdot_x = diff(f,x);
    fdot_x0 = subs(fdot_x,x0);
    if fdot_x0<0
```

```
            xk1 = x0+2^k*Step_Size;
        elseif fdot_x0>0
            xk1 = x0-2^k*Step_Size;
        end
        k=k+1;
        fxk1 = subs(f,xk1);
        fdot_xk1 = subs(fdot_x,xk1);
      end
      %x1=x0;
      %x2=xk1;
    while(abs(fxbardot)>=Tolerance)
      fx1=subs(f,x0);
      fx2=subs(f,xk1);
      fx1dot=subs(fdot_x,x0);
      fx2dot=subs(fdot_x,xk1);
      z=(3*(fx1-fx2)/(xk1-x0)) + fx2dot +fx1dot;
      w = ((xk1-x0)/abs(xk1-x0))*(z^2 - (fx1dot*fx2dot))^(1/2);
      mu = (fx2dot + w - z)/(fx2dot - fx1dot + 2*w);
      if(mu<0)
          x_bar=xk1;
      elseif(0<=mu && mu<=1)
          x_bar=xk1 - mu*(xk1-x0);
      else
          x_bar=x0;
      end
      fxbar = subs(f,x_bar);
      fxbardot = subs(fdot_x,x_bar);
      fx1=subs(f,x0);
      while(fxbar>fx1)
          x_bar = x_bar - ((x_bar - x1)/2);
          fxbar = subs(f,x_bar);
          fxbardot = subs(fdot_x,x_bar);
          fx1=subs(f,x0);
          fx1dot=subs(fdot_x,x0);
      end
      fxbar = subs(f,x_bar);
      fxbardot = subs(fdot_x,x_bar);
      if(abs(fxbardot)<=Tolerance  && abs((x_bar-x1)/x_bar)<=E_2)
              break;
      elseif(fxbardot*fx1dot < 0)
          xk1=x0;
          x0=x_bar;
      elseif(fxbardot*fx2dot < 0)
          x0=x_bar;
      end
    m=m+1;
    end
    m
    vpa(x_bar)
    vpa(fxbar)
  else
      fprintf('\nPlease Enter a Valid Input (1-5)');
  end
  true = input('\n\nEnter \nContinue:1 \nExit:0 \nEnter Option: ');
  end
```

RESULTS

Table1: Results using given Error Tolerance

METHOD	NUMBER OF LOOPS	ERROR TOLERANCE	MINIMUM	X AT MINIMUM
Interval Halving	22	1e-05	7.69601	2.50747
Golden Section	31	1e-05	7.69601	2.50747
Bisection	21	1e-05	7.69601	2.50747
Powell's Method	6	1e-05	7.69601	2.50747
Cubic Search Method	5	1e-05	7.69601	2.50747

Table2: Sensitivity to termination criterion – error tolerance 1e-04

METHOD	NUMBER OF LOOPS	MINIMUM	X AT MINIMUM
Interval Halving	18	7.69601	2.50747
Golden Section	26	7.69601	2.50747
Bisection	18	7.69601	2.50747
Powell's Method	5	7.69601	2.50618
Cubic Search Method	4	7.69601	2.50747

Table3: Sensitivity to termination criterion – error tolerance 1e-03

METHOD	NUMBER OF LOOPS	MINIMUM	X AT MINIMUM
Interval Halving	15	7.69601	2.50732
Golden Section	22	7.69601	2.50746
Bisection	10	7.69601	2.50732
Powell's Method	5	7.69602	2.50618
Cubic Search Method	3	7.69601	2.50750

Table4: Sensitivity to termination criterion – error tolerance 1e-02

METHOD	NUMBER OF LOOPS	MINIMUM	X AT MINIMUM
Interval Halving	12	7.69601	2.50732
Golden Section	17	7.69601	2.50785
Bisection	10	7.69601	2.50732
Powell's Method	4	7.69648	2.49528
Cubic Search Method	2	7.69601	2.50784

OBSERVATIONS AND CONCLUSIONS

Number of loops (execution time)

1. Cubic search method is the fastest method - it utilizes only loops.

2. The fastest Method is Bisection Method. Cubic Search Method takes the most time.

3. Powell's method is the optimal method - takes fewer loops and agreeable execution time.

Sensitivity to termination criteria

The sensitivity was checked for four different error tolerance values:

1. 1e-05 : Initial tolerance value, all values are similar up to 5 decimal places
2. 1e-04 : Minimum and x at minimum do not vary up to 5 decimal places for all methods, minimum value changes slightly in Powell's method (within 5 decimal places).
3. 1e-03 : Golden Section and Cubic Search x values are close, rest of the methods show noticeable variation in x within 5 decimal places, minimum value changes slightly in Powell's method (within 5 decimal places).
4. 1e-02 : Cubic Search stays accurate in minimum value while the rest of the methods show variations for x within 5 decimal places and the minimum values vary (within 5 decimal places).

Number of function evaluations (Original and derivatives)

1. Interval halving, Golden section and Powell's Sequential Quadratic methods do not employ derivatives in calculating the minimum.

2. As seen in the code, Bisection method and Cubic Search algorithms utilize the derivative of the function.

3. Bisection method utilizes the least number of original and derivative function evaluations.

4. Interval halving utilizes the maximum number of original function evaluations.

HONOR CODE

I have done the work on my own and have not received any help.

SNEHA GANESH