SNEHA GANESH

ME 8710 ENGINEERING OPTIMIZATION

PROBLEM STATEMENT

Test the methods on the following function:

- Simplex Search Method
- Powell's Conjugate Method
- Bisection (1st order)
- Conjugate Gradient Method

Function:

$f(x1,x2,x3,x4) = (4x_1 + 2x_2 - x_3 + 2x_4)^2 + (x_1 - 3x_2 + x_3 - 3x_4)^2 + (-x_1 + x_2 + 3x_3)^2$

Set a termination criterion (ε <= 0.0001), input the initial values as X0=$(1,1,2,1)^T$ and step size of 1.

MATLAB CODE

```
%UNCONSTRAINED MULTIVARIABLE N DIMENSIONAL MINIMIZATION/OPTIMIZATION METHODS

clc;
true=1;
while true
clearvars
syms x1 x2 x3 x4 x5;
syms f a X Y;
M = [x1 x2 x3 x4 x5];
iter=1; error=1;

fprintf('\nUNCONSTRAINED MULTIVARIABLE N DIMENSIONAL
MINIMIZATION/OPTIMIZATION METHODS')
fprintf('\n 1.SIMPLEX SEARCH METHOD \n 2.POWELLS CONJUGATE METHOD')
fprintf('\n 3.STEEPEST DESCENT METHOD \n 4.CONJUGATE GRADIENT METHOD \n')
ch = input('\nChoose Method: ');
n = input('\nEnter Dimension (1 to 5): ');
X= zeros(1,n,'sym');
for p=1:1:n
    X(1,p)= M(1,p);
end
f = input('\nEnter Function, f(x1,x2..,xn): ');
Tolerance = input('\nEnter Tolerance: ');
X_0 = input('\nEnter Initial values in the form "[x1 x2 .. xn]": ');
```

SIMPLEX SEARCH METHOD

```
if(ch==1)
%SIMPLEX SEARCH METHOD
tic
Y=zeros();
f_x = zeros();
f_x(1,1) = subs(f,X,X_0);
I = zeros(n,n);
iter=0;
for j=1:1:n
    I(j,j)=1;
end
a = input('\nEnter step size: ');
p=((a/(n*sqrt(2)))*(((sqrt(n+1))+(n-1))));
q=((a/(n*sqrt(2)))*((sqrt(n+1))-1));

%SIMPLEX SEARCH
  for j=1:1:n
     ej=I(:,j);
     for k=1:1:n
       if(k~=j)
         ek=I(:,k);
         Y = Y + q*ek;
       end
     end
     Y = X_0.' + p*ej + Y;
     S(:,j+1) = Y;
     f_x(1,j+1) = subs(f,X,vpa(Y.'));
  end
  R = sort(f_x(1,:));
  [rows column] = size(R);
  fx_h = R(1,column);
  fx_l = R(1,1);
  fx_s = R(1,column-1);
  S(:,1) = X_0.';

 for q=1:1:column
     f_c = double(subs(f,X,S(:,q).'));
     if(f_c == fx_h)
       x_h(:,1) = S(:,q);
       xh_s = q;
     elseif(f_c == fx_l)
         x_l(:,1) = S(:,q);
         xl_s = q;
     elseif(f_c == fx_s)
         x_s(:,1) = S(:,q);
         xs_s = q;
     end
end
  x_bar = ((1/n)*((S(:,1)+ S(:,2)+ S(:,3)+ S(:,4)+ S(:,5) - S(:,xh_s))));
  fx_bar=subs(f,X,x_bar.');
  alpha=1;
  beta=2;
  gamma=0.5;
```

```
while(error >= Tolerance)

  x_r = double(x_bar + alpha*(x_bar - x_h));
  fx_r=double(subs(f,X,x_r.'));

 if(fx_l < fx_r && fx_r <= fx_s)
      x_h=x_r;
      S(:,xh_s)=x_h;
      fx_h=subs(f,X,x_h.');
      R(1,column)=fx_h;
      R =sort(R);
      fx_h = R(1,column);
      fx_l= R(1,rows);
      fx_s = R(1,column-1);
 elseif(fx_r < fx_l)
      x_e= double(x_bar + beta*(x_r - x_bar));
      fx_e=double(subs(f,X,x_e.'));
      if(fx_e < fx_r)
          x_h=x_e;
          fx_h=subs(f,X,x_h.');
          R(1,column)=fx_h;
          S(:,xh_s)=x_e;
          R=sort(R);
          fx_h = R(1,column);
          fx_l= R(1,rows);
          fx_s = R(1,column-1);
          %continue
      else
          x_h=x_r;
          S(:,xh_s)=x_h;
          fx_h=subs(f,X,x_h.');
          R(1,column)=fx_h;
          R=sort(R);
          fx_h = R(1,column);
          fx_l= R(1,rows);
          fx_s = R(1,column-1);
          %continue
      end
 elseif(fx_r >= fx_h)
      x_c = x_bar + gamma*(x_h - x_bar);
      fx_c = subs(f,X,x_c.');
      if(fx_c > fx_h)
          for q=2:1:column
              if(subs(f,X,S(:,q).')== fx_l)
                  x_l = S(:,q);
              elseif(subs(f,X,S(:,1).')== fx_l)
                  x_l = S(:,1);
              end
          end
          for q=2:1:column
               S(:,q) = S(:,q) + 0.5*(x_l - S(:,q));
          end
          for q=1:1:column-1
          f_x(1,q) = subs(f,X,(S(:,q+1).'));
          end
          R = sort(f_x(1,:));
          fx_h = R(1,column);
```

```
            fx_l= R(1,rows);
            fx_s = R(1,column-1);
        else
           x_h = x_c;
           fx_h=subs(f,X,x_c.');
           R(1,column)=fx_h;
           S(:,xh_s)=x_c;
           for q=1:1:column
           f_x(:,q) = subs(f,X,S(:,q).');
           end
           R = sort(f_x);
           fx_h = R(1,column);
           fx_l = R(1,1);
           fx_s = R(1,column-1);
        end

    elseif(fx_r<fx_h)
        x_h = x_r;
        x_c = x_bar + gamma*(x_h - x_bar);
        if(fx_c > fx_h)
            for q=2:1:column
                if(subs(f,X,S(:,q).')== fx_l)
                    x_l = S(:,q);
                elseif(subs(f,X,S(:,1).')== fx_l)
                    x_l = S(:,1);
                end
            end
            for q=2:1:column
                 S(:,q) = S(:,q) + 0.5*(x_l - S(:,q));
            end
            for q=1:1:column-1
            f_x(1,q) = subs(f,X,(S(:,q+1).'));
            end
            R = sort(f_x(1,:));
            fx_h = R(1,column);
            fx_l= R(1,rows);
            fx_s = R(1,column-1);
        else
           x_h = x_c;
           fx_h=subs(f,X,x_c.');
           R(1,column)=fx_h;
           S(:,xh_s)=x_c;
           for q=1:1:column
           f_x(:,q) = subs(f,X,S(:,q).');
           end
           R = sort(f_x);
           fx_h = R(1,column);
           fx_l = R(1,1);
           fx_s = R(1,column-1);
        end
    end
    for q=1:1:column
        f_c = double(subs(f,X,S(:,q).'));
        if(f_c == fx_h)
          x_h(:,1) = S(:,q);
          xh_s = q;
        elseif(f_c == fx_l)
```

```
        x_l(:,1) = S(:,q);
        xl_s = q;
    elseif(f_c == fx_s)
        x_s(:,1) = S(:,q);
        xs_s = q;
    end
end
  x_bar = ((1/n)*((S(:,1) + S(:,2)+ S(:,3)+ S(:,4)+ S(:,5) - S(:,xh_s))));
  fx_bar=double(subs(f,X,x_bar.'));
  er = double((1/(n+1))*(double((subs(f,X,S(:,1).')- fx_bar)^2 +
(subs(f,X,S(:,2).')- fx_bar)^2 + ((subs(f,X,S(:,3).')- fx_bar)^2 +
(subs(f,X,S(:,4).')- fx_bar)^2 + (subs(f,X,S(:,5).')- fx_bar)^2))));
  error = double(sqrt(er));
  iter=iter+1;
end
t=toc;
z= double(fx_bar);
Z= double(-1*x_bar.');
fprintf('\n Number of iterations: %d \n Function Minimum: %.12f ',iter,z)
fprintf('\n Function Time: %.8f \n Values of Variables at Function Minimum :
%.12f,%.12f,%.12f,%.12f \n',t,Z)
```

POWELLS CONJUGATE METHOD

```
elseif(ch == 2)

%POWELLS CONJUGATE METHOD
tic
alpha= -1; m=0;
f_x(2,1) = 1; f_min = 1; %f_x(1,2) = 2; f_x(1,3) =1;
f_x(1,1) = subs(f,X,X_0);
I = zeros(n,n);
iter=1;
for j=1:1:n
    I(j,j)=1;
end
S_p = I(:,1);
P = X_0;
while (error > Tolerance)
 while(alpha < sqrt((f_x(1,1) - f_x(2,1))/(f_min)))
   X_0 = P
   f_x(1,1) = subs(f,X,X_0)
   K = double(X_0);
  for k=1:1:n
   X_a = X_0.'+ a*(I(:,k))
   Q = X_a.';
   f_a = subs(f,X,Q)
   alpha = double(solve(diff(f_a,a)==0))
   Y = double(X_0.' + alpha*(I(:,k)))
   R = Y';
   S(:,k+1)=Y;
   f_x(1,k+1) = subs(f,X,R)
   %vpa(f_x(1,k+1));
```

```
   X_0 = R;
  end
 S(:,1)= K.';
 S_p = Y - K.'
  if(S_p~=zeros())
   X_a = K.'+ a*S_p
   Q = X_a.';
   f_a = subs(f,X,Q)
   alpha = double(solve(diff(f_a,a)==0))
   Y_x = double(K.' + alpha*S_p)
   P = Y_x';
   f_x(2,k+1-n) = double(subs(f,X,P))
   iter=iter+1;
   for i=1:1:k
     f_dec(1,i) = f_x(1,i) - f_x(1,i+1);
   end
   f_min = max(f_dec)
  end
 error=double(norm(S_p));
 end
 if(error > Tolerance && alpha > ((f_x(1,1) - f_x(2,1))/(f_x(1,2)-
f_x(1,3))^(1/2)))
   for i=1:1:k
    if(f_min == (subs(f,X,(S(:,k)).') - subs(f,X,(S(:,k+1)).')))
        I(:,k) = S_p;
    end
   end
   %f_x(1,1) = subs(f,X,X_0);
   alpha=-1;
   %f_x(2,1) = 1; f_x(1,2) = 2; f_x(1,3) =1;
   iter=iter+1;
 end
m=m+1;
end

%From multiple runs and observations Powell's conjugate comes twice as close
to values as that of the gradient methods for the given function,
%that is, P(:,1:4)= 0.5*P(:,1:4); z = double(subs(f,X,P));

t=toc;
z= double(f_x(2,k+1-n));
Z= double(P);
fprintf('\n Number of iterations: %d \n Number of cycles:  %d \n Function
Minimum: %.16f ',iter,m,z)
fprintf('\n Function Time: %.8f \n Values of Variables at Function Minimum :
%.12f,%.12f,%.12f,%.12f \n',t,Z)
```

STEEPEST DESCENT METHOD

```
elseif(ch==3)

%STEEPEST DESCENT METHOD
tic
j=0;
for i=1:1:n
fdot_x0(1,i)=diff(f,X(1,i));
end
fdot_x0
f_xdot = subs(fdot_x0,X,X_0)
while(error > Tolerance)
    X_a = X_0.' - a*f_xdot.';
    Y = X_a.';
    f_y = subs(f,X,Y);
    f_y_dot=diff(f_y,a);
    alpha = double(solve(f_y_dot==0));
    %double(alpha);
    Q = double(X_0.' - alpha*f_xdot.');
    X_0 = Q.';
    f_xdot = double(subs(fdot_x0,X,X_0));
    error = double(norm(f_xdot)./(1 + (abs(norm(f_xdot)))));
    j=j+1;
end
t=toc;
j;
error;
z=double(subs(f,X,X_0));
Z=double(X_0);
fprintf('\n Number of iterations: %d \n Function Minimum: %.12f ',j,z)
fprintf('\n Function Time: %.8f \n Values of Variables at Function Minimum :
%.12f,%.12f,%.12f,%.12f \n',t,Z)
```

CONJUGATE GRADIENT METHOD

```
elseif(ch==4)

%CONJUGATE GRADIENT METHOD
tic
j=0;
for i=1:1:n
fdot_x0(1,i)=diff(f,X(1,i));
end
fdot_x0
f_xdot_0 = subs(fdot_x0,X,X_0)
while(error > Tolerance)
    X_a = X_0.' - a*f_xdot_0.';
    Y = X_a.';
    f_y = subs(f,X,Y);
    f_y_dot=diff(f_y,a);
    alpha = solve(f_y_dot==0);
    %double(alpha)
```

```
    Q = X_0.' - double(alpha)*f_xdot_0.';
    %double(Q);
    X_0 = Q.';
    f_xdot = double(subs(fdot_x0,X,X_0));
    %(f_xdot);
    beta = (norm(f_xdot))^2/(norm(f_xdot_0))^2;
    %double(beta);
    S = double(-1*(f_xdot) + beta*(-(f_xdot_0)));
    %(S);
    error = abs(double(norm(f_xdot)));
    f_xdot_0 = S;
    j=j+1;
end

t=toc;
j;
z=double(subs(f,X,X_0));
Z=double(X_0);


fprintf('\n Number of iterations: %d \n Function Minimum: %.12f ',j,z)
fprintf('\n Function Time: %.8f \n Values of Variables at Function Minimum :
%.12f,%.12f,%.12f,%.12f \n',t,Z)

else
    fprintf('\nPlease Enter a Valid Input (1-4)');
end
true = input('\n\nEnter \nContinue:1 \nExit:0 \nEnter Option: ');
if(true==0)
    clc;
end
end
```

RESULTS

Table1: Results using given Error Tolerance

METHOD	NUMBER OF ITERATIONS	TIME (SEC) (Fastest run)	MINIMUM	X AT MINIMUM
Simplex Search	67	16.97869599	0.000019321760	0.003182249815, -0.167385239317, 0.056353846482, 0.188534139333
Powell's Conjugate	Cycles:2 11	1.45946228	0.000000000003	0.016611549439, -0.681073363189, 0.232561634748, 0.764131088274
Steepest Descent	25	1.44710968	0.000000000205	0.008514683380, -0.349023098911 0.119183157251 0.391586392603
Conjugate Gradient	21	1.26714219	0.000000000223	0.008515084403 -0.349021916468 0.119183099478 0.391587455385

OBSERVATIONS AND CONCLUSIONS

Number of iterations (and execution time)

1. Conjugate Gradient method is the fastest method (Execution Time: 1.26714219 sec).

2. The Simplex Search Method takes the most time and the maximum number of loops. (From observations, values are scaled down by around 2.5 times and accuracy of function minimum is compromised).

3. Powell's method shows optimal execution - takes fewer loops and faster execution time.

4. Powell's Conjugate method, as expected to reach convergence before the N step or before, reaches function minimum at the second cycle.

Number of function evaluations (Original and derivatives)

1. Simplex Search Method and Powell's Conjugate methods do not employ derivatives in calculating the minimum.

2. Steepest Descent and Conjugate Gradient algorithms utilize the derivative of the function.

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

4. Simplex Search utilizes the maximum number of original function evaluations.

Sensitivity to Error Tolerance:

1. Varying the tolerance below given value decreases number of iterations, X values do not change dramatically (changes are observed in decimal places greater than 5) but function minimum varies noticeably for each (power 1) decrease in tolerance. Function time also decreases by a small margin.

2. Varying the tolerance above given value increases number of iterations, X values do not change dramatically (changes are observed in decimal places greater than 5) but function minimum decreases noticeably. Function time also increases by a small margin.

HONOR CODE

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

SNEHA GANESH