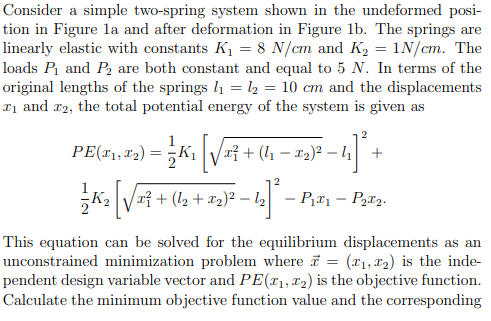
2021

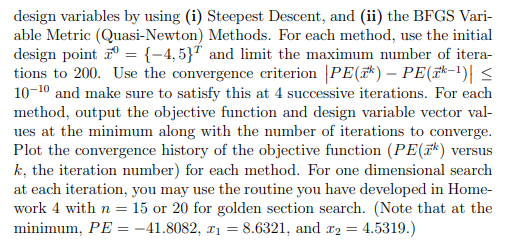
# HW3

AERO 5830 - S. HOSDER

MATT PAHAYO

Question 1





**Results**

****

**Chart, line chart

Description automatically generated**

**Methodology**

Get the search direction from the gradient. Equation 1 will give a unit vector and is for the Steepest Descent Method.

(1)

Get alpha star by evaluating the function at the guess plus the search direction times alpha. Then find the minimum value of alpha by minimizing the alpha-function with a Golden Section algorithm and then fit a cubic function that is outputted by the Golden Section algorithm to get alpha star.

(2)

(3)

Everything on the right-hand side is known, so get the next iteration of x-vector. These steps are repeated until the stopping criteria is met (equation 4).

(4)

To use the BFGS algorithm all that needs to be changed is how the search direction is calculated. This is done by approximating the Hessian matrix at xk and multiplying it by the negative of the gradient of the function at xk.

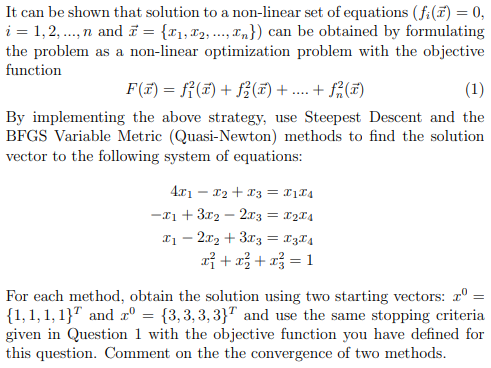
(5)

To approximate the Hessian, let the first iteration be the identity matrix at the first iteration and to get the next define D.

Text, letter

Description automatically generated

**Question 2**



**Results**

****

The Steepest Descent converges to a value for both guesses while BFGS does not seem to do so. The Descent converges to zero while the BFGS does not in the given number of iterations.

Methodology

To solve the system, the functions are first rearranged, so the functions equal zero. Then a function is defined where it is the L2norm of the other functions. The algorithms developed in question 1 are then used on this new function.

(6)

classdef mOpt

% mOpt is a multivariable optimizer

methods (Static)

function [Fq,q,iter,PE] = Steep(F,q0,tol)

syms x1 x2 x3 x4 as

% Steep is a function that utilizes the

% steepest descent method

q = q0;

q1(1) = q0(1);

q2(1) = q0(2);

q3(1) = q0(3);

q4(1) = q0(4);

i=2;

q1(2) = 10^99;

q2(2) = 10^99;

q3(2) = 10^99;

q4(2) = 10^99;

gradF(x1,x2,x3,x4) = gradient(F,[x1,x2,x3,x4]);

s = @(x1,x2,x3,x4) -gradF(x1,x2,x3,x4)/norm(gradF(x1,x2,x3,x4));

while abs(F(q1(i),q2(i),q3(i),q4(i))-F(q1(i-1),q2(i-1),q3(i-1),q4(i-1)))>tol

if q1(2) == 10^99

q1(2) = q0(1);

q2(2) = q0(2);

q3(2) = q0(3);

q4(2) = q0(4);

end

search = double(s(q1(i),q2(i),q3(i),q4(i)));

fas(as) = F(q1(i)+search(1)\*as,q2(i)+search(2)\*as,q3(i)+search(3)\*as,q4(i)+search(4)\*as);

[xlow,w2,w1,xhigh] = onedOpt.Gold(0,2,20,0,-fas);

[alpha,y] = cubicFit(xlow,w2,w1,xhigh,-fas);

q = q + alpha(1)\*search

PE(i-1,1) = F(q1(i),q2(i),q3(i),q4(i));

q1(i+1) = q(1);

q2(i+1) = q(2);

q3(i+1) = q(3);

q4(i+1) = q(4);

i = i + 1;

if i == 201

break;

end

end

Fq = F(q1(end),q2(end),q3(end),q4(end));

iter = i - 1;

end

function [Fq,q,iter,PE] = BFGS(F,q0,theta,tol)

%METHOD1 Summary of this method goes here

% Detailed explanation goes here

syms x1 x2 x3 x4 as

q = q0;

q1(1) = q0(1);

q2(1) = q0(2);

q3(1) = q0(3);

q4(1) = q0(4);

i=2;

q1(2) = 10^99;

q2(2) = 10^99;

q3(2) = 10^99;

q4(2) = 10^99;

gradF(x1,x2,x3,x4) = gradient(F,[x1,x2,x3,x4]);

H = eye(length(q));

D = 0;

while abs(F(q1(i),q2(i),q3(i),q4(i))-F(q1(i-1),q2(i-1),q3(i-1),q4(i-1)))>tol

if q1(2) == 10^99

q1(2) = q0(1);

q2(2) = q0(2);

q3(2) = q0(3);

q4(2) = q0(4);

end

if i >2

p = [q1(i)-q1(i-1);q2(i)-q2(i-1);q3(i)-q3(i-1);q4(i)-q4(i-1)];

y = double(gradF(q1(i),q2(i),q3(i),q4(i))-gradF(q1(i-1),q2(i-1),q3(i-1),q4(i-1)));

sigma = transpose(p)\*y;

tau = transpose(y)\*H\*y;

D = (sigma+theta\*tau)/sigma^2\*p\*transpose(p)...

+(theta-1)/tau\*H\*y\*transpose(H\*y)-...

theta/sigma\*(H\*y\*transpose(p)+p\*transpose(H\*y));

end

H = H + D;

search = double(-H\*gradF(q1(i-1),q2(i-1),q4(i-1),q3(i-1))/norm(-H\*gradF(q1(i-1),q2(i-1),q3(i-1),q4(i-1))));

fas(as) = F(q1(i)+search(1)\*as,q2(i)+search(2)\*as,q3(i)+search(3)\*as,q4(i)+search(4)\*as);

[xlow,w2,w1,xhigh] = onedOpt.Gold(0,2,20,0,-fas);

[alpha,y] = cubicFit(xlow,w2,w1,xhigh,-fas);

q = q + alpha(1)\*search;

PE(i-1,1) = F(q1(i),q2(i),q3(i),q4(i));

q1(i+1) = q(1);

q2(i+1) = q(2);

q3(i+1) = q(3);

q4(i+1) = q(4);

i = i + 1;

if i == 201

break;

end

end

Fq = F(q1(end),q2(end),q3(end),q4(end));

iter = i - 1;

end

end

end

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 it 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

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

% main.m

clc

clear all

close all

format longg

syms x1 x2 x3 x4 as

q1 = -4;

q2 = 5;

q3 = 0;

q4 = 0;

q0 = [q1;q2;q3;q4];

tol =10^-10;

k1 = 8;

k2 = 1;

P1 = 5;

P2 = 5 ;

l1 =10;

l2 = 10;

theta = 1;

F = @(x1,x2,x3,x4) 1/2\*k1\*(sqrt(x1^2+(l1-x2)^2)-l1)^2+k2/2\*(sqrt(x1^2+(l2+x2)^2)-l2)^2-P1\*x1-P2\*x2;

[Fq,q,iter,PE] = mOpt.BFGS(F,q0,theta,10^-6)

[Fs,qs,iters,PEs] = mOpt.Steep(F,q0,10^-6)

hold on

plot(1:length(PEs),PEs)

plot(1:length(PE),PE)

xlabel('Iterations')

legend('Steepest Descent','BFGS')

ylabel('PE')

title('Convergence history of the objective function')

grid on

x0 = [1;1;1;1]

F = @(x1,x2,x3,x4) (4\*x1-x2+x3-x1\*x4)^2+ ...

(-x1+3\*x2-2\*x3-x2\*x4)^2 + (x1-2\*x3+3\*x3-x3\*x4)^2 + ...

(x1^2+x2^2+x3^2-1)^2

[Fq,q,iter,PE] = mOpt.BFGS(F,x0,theta,tol)

[Fq,q,iter,PE] = mOpt.Steep(F,x0,tol)