% Course: Nonlinear Optimization. %

% FALL.2018. Dr. Cheng. %

% Assignment: (3) %

% Date:(2018.10.10) %

% By: 卢博 %

% ID NUMBER: 11849159 %

% LAB:(4) %

% Description: This code uses the nor\_path and min\_path function to find a shorter way.

1. Golden Ratio Search

clear;clc  
x1 = 0;x2 = 10;  
f1 = @(x) (x - 2).^2;  
x3 = 0;x4 = 2\*pi;  
f2 = @(x) cos(x);  
[x1,x2] = Golden\_Ratio\_Search(f1,[x1,x2]);  
[x3,x4] = Golden\_Ratio\_Search(f2,[x3,x4]);  
  
x\_f1 = (x1 + x2)/2  
x\_f2 = (x3 + x4)/2

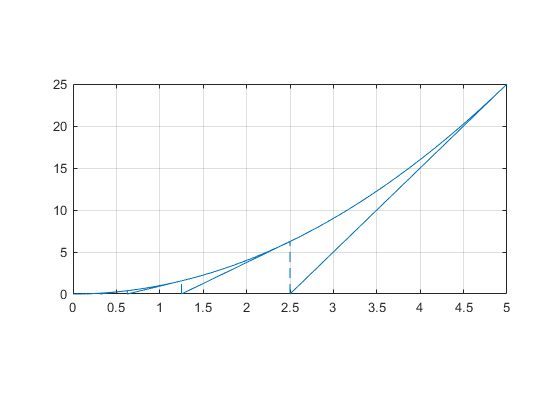
x\_f1 =  
 2.0002  
x\_f2 =  
 3.1416

function [y1,y2] = Golden\_Ratio\_Search(f,x)  
t = (1 + sqrt(5))/2;  
x1 = x(1);x2 = x(2);  
  
while abs(x1 - x2) > 1e-3  
 x3 = x2 - (x2 - x1)/t;  
 x4 = x1 + (x2 - x1)/t;  
 fy = [f(x1) f(x2) f(x3) f(x4)];  
 idx = find(fy == max(fy));  
 if idx == 1  
 x1 = x3;  
 elseif idx == 2  
 x2 = x4;  
 elseif idx == [1,2]  
 x1 = x3;  
 x2 = x4;  
 end  
end  
  
y1 = x1;  
y2 = x2;  
end

1. Newton’s Method

clear;clc  
syms x  
x1 = 5;  
f1 = x^2;  
fplot(f1,[0 x1])  
set(gca,'position',[0.1300 0.300 0.7750 0.5])  
grid on  
hold on  
x1 = vpa(Newtons\_Method(f1,x1))

x1 =  
0.0006103515625



function y = Newtons\_Method(f,x)  
% 直接带入公式即可  
while abs(subs(f,x)) >= 1e-6  
 fx = subs(diff(f),x);  
 newx = x - subs(f,x)/fx;  
 line([newx x],[0 subs(f,x)])  
 line([newx newx],[0 subs(f,newx)],'linestyle','--');  
 x = newx;  
end  
y = x;  
end

1. Descent Method with Trust Region

clear;clc;  
syms x x1 x2;  
x10 = [5;3];  
f1 = exp(x1/5 + x2/2) + x1^2 + x2^2;  
x20 = 2;  
f2 = cos(x);  
  
[ans1,iter1] = Trust\_Region(f1,[x1;x2],x10,1,0.25,0.75,1e-3);  
final\_f1 = double(subs(f1,[x1;x2],ans1))  
[ans2,iter2] = Trust\_Region(f2,[x],x20,1,0.25,0.75,1e-3);  
final\_f2 = double(subs(f2,ans2))

final\_f1 =  
 0.9363  
final\_f2 =  
 -1

function[x, iter] = Trust\_Region(f, var, x0, r0, miu, yita, eps)  
% 目标函数: f  
% 自变量向量: var  
% 初始点: x0  
% 初始信赖域半径: r0  
% 初始参数: miu  
% 初始参数: yita  
% 精度: eps  
% 目标函数取最小值的自变量值: x  
% 迭代次数: iter  
tol = 1;  
r = r0;  
x = x0;  
iter = 1; % 迭代次数  
while tol > eps  
 jacf = jacobian(f, var); % 雅可比矩阵  
 fx = double(subs(f, var, x)); % 计算目标函数值  
 v = subs(jacf, var, x); % 计算雅克比矩阵值  
 tol = double(norm(v));  
 M1 = double(transpose(v)); % 二次规划 中的一次项矩阵  
 lb = -r \* ones(length(var), 1); % r为信赖域半径，自变量下界约束  
 ub = r \* ones(length(var), 1); % 自变量上界约束  
 [y, fy] = quadprog([], M1, [], [], [], [], lb, ub); % 求解二次规划  
 fx\_n = double(subs(f, var, x + y)); % 重新计算目标函数值  
 p = (fx - fx\_n) / (-fy); % 计算目标函数实际下降与预测下降之比  
 if p <= miu % 目标函数实际下降不明显  
 r = 0.5 \* r;  
 else % 目标函数值下降明显  
 x = x + y; % 更新参数, 扩大信赖域半径  
 if p >= yita % 如果 p > yita  
 r = 2 \* r;  
 end  
 end  
 iter = iter + 1;  
end  
end