%%这一块是test模块(test代码开始)

% Numerical Optimization

% Name gyj? Student number19020241153846

%

% This function is the entry of the program

%

% Answer the questions in the assignments one by one

% e.g.,

% 1, As shown in Figure 1, I found ...

% As indicated in the output, I concluded that ...

% 2.1 It can be seen from Figure 2 that ....

% 2.2 We can observe .... That is because .... My conjecture is ....

% ...

seed = 3846; %"the last four digits of your student number";

fprintf('seed:%d\n', seed);

rng(seed);

% default params

params.alpha = 1e-4;

params.maxiter = 50000;

params.verbose = 2;

params.lsmaxiter = 30;

params.ratio = 0.25;

params.OutputGap = 10;

% For initial iterate, we need to specify its dimension first.

n = 10;

x0.main = randn(n, 1);

params.x0 = x0; % initial iterate

% For the first problem

n = 10;

D = 1:n;

O = orth(randn(n, n));

A = O \* diag(D) \* O';

eigv = eig(A);

connum = max(eigv) / min(eigv);

b = randn(n, 1);

x0.main = randn(n, 1);

fns = [];

fns = ProbEucQuadratic(A, b, fns);

fprintf('\nSteepest descent method for the first problem!\n');

[xopt, info] = RSD(fns, params);

%%test代码结束

%%这里是problems模块(problems代码开始)

function fns = ProbEucQuadratic(A, b, fns)

% Define the quadratic convex problem : f(x) = 0.5 x^T A x - b^T x, A is SPD

%

% INPUT:

% A : a square matrix, SPD

%

% OUTPUT:

% fns : a structure

% fns.f : the function handle of the objective function

% fns.Grad: the function handle of the gradient

%

%

% By Wen Huang

fns.f = @(x)f(x, A, b);

fns.Grad = @(x)Grad(x, A, b);

fns.HessEta = @(x, v)HessEta(x, v, A, b);

end

function [output, x] = f(x, A, b)

x.Ax = A \* x.main;

output = 0.5 \* x.main' \* x.Ax - b' \* x.main;

end

function [output, x] = Grad(x, A, b)

....

end

function [output, x] = HessEta(x, v, A, b)

....

end

%%problems代码结束

%%这里是solvers第一个函数代码

function [eta2, stepsize, x2, f2, gradf2, slopex2, LSinfo, status] = linesearch(eta1, x1, fvs, gradf1, initslope, initstepsize, iter, fns, params)

% This function apply the backtracking algorithm to find an appropriate step size.

%

% INPUT:

% eta1 : search direction

% x1 : current iterate

% fvs : previous function values, the last one is the function value at the current iterate

% gradf1 : gradient at the current iterate

% initslope : the initial slope

% initstepsize : the initial step size

% fns : a struct that contains required function handles

% fns.f(x) : return objective function value at x.

% fns.Grad(x) : return the gradient of objection function at x.

% params: : a struct that contains parameters that are used

% alpha : the coefficient in the Armijo condition

% ratio : the shrinking parameter 0 < ratio < 1

% lsmaxiter : the max numbers of iteration in the backtracking algorithm

%

% OUTPUT:

% eta2 : step size \* search direction

% stepsize : desired step size

% x2 : next iterate x + eta2

% f2 : function value at the next iterate

% gradf2 : gradient at the next iterate

% slopex2 : slope at the accepted step size

% LSinfo : Debug information in line search algorithm

% LSinfo.lf : the number of function evaluations

% LSinfo.lgf: the number of gradient evaluations

% status : 0 means success

% 1 means line search fails with inner iterations reaches its max number of iterations

% 2 means the numerical errors dominates the computation

%

% By Wen Huang

f1 = fvs(end); % function value at current iterate

stepsize1 = initstepsize;

eta2 = stepsize1 \* eta1;

....

status = 0; % line search success by default

while(....)

....

btiter = btiter + 1;

end

....

if(btiter >= params.lsmaxiter) % line search fails

fprintf('warning: line search fails at iter:%d!\n', iter);

status = 1;

end

if(norm((x1.main + eta2) - x1.main) == 0)

status = 2;

end

LSinfo.lf = xxx; LSinfo.lgf = xxx;

end

%%solvers第一个函数代码结束

%%这里是solvers第二个函数代码

function [xopt, info] = RSD(fns, params)

% Solver of the steepest descent method

%

% INPUT:

% fns : a struct that contains required function handles

% fns.f(x) : return objective function value at x.

% fns.Grad(x) : return the gradient of objection function at x.

%

% params : a struct that contains parameters that are used in the solver.

% params.x0 : initial approximation of minimizer.

% params.maxiter : the maximum number of iterations

% params.verbose : '0' means silence, '1' means output information of initial and final iterate.

% '2' means output information of every iterate.

%

%

% OUTPUT:

% xopt : the last iterate

% info : informtion generated during the algorithms

%

%

% By Wen Huang

x1 = params.x0;% initial iterate

[f1, x1] = fns.f(x1); % function value

[gradf1, x1] = fns.Grad(x1); % Riemannian gradient

....

fvs = f1; ngfs = ngf1; lf = 1; lgf = 1; % information

status = 0;

while(....)

% Get search direction

....

% Find appropriate initial step size

....

[eta2, stepsize, x2, f2, gradf2, slopex2, LSinfo, status] = LinesearchBacktracking(eta1, x1, fvs, gradf1, initslope, initstepsize, iter, fns, params);

....

% Line search algorithm

....

% finish line search

fvs(end + 1) = f2;

ngfs(end + 1) = ngf2;

iter = iter + 1;

lf(iter) = LSinfo.lf; lgf(iter) = LSinfo.lgf;

% Get ready for the next iteration

x1 = x2; f1 = f2; gradf1 = gradf2; ngf1 = ngf2;

%output

....

end

if(params.verbose >= 1)

fprintf('iter:%d,f:%.3e,|gf|:%.3e,|gf|/|gf0|:%.3e,lf:%d,lgf:%d,status:%d\n', iter, f1, ngf1, ngf1 / ngf0, sum(lf), sum(lgf), status);

end

xopt = x1;

info.iter = iter;

info.ngfs = ngfs;

info.fvs = fvs;

info.lf = lf;

info.lgf = lgf;

end

%%solvers第二个函数代码结束