MHCG CODE.

%function STCG(fnum,dimnum,xnum,xlrange,tol,maxit,maxfev)

% A Dai Liao type projection algorithm for convex constraints monotone equations

% with applications in compressive sensing

% A. B. Abubakar, H. Mohammad and P. Kumam March 30 2018

% Global convergence method

% call: dlcs(f,x0,tol,maxit)

% Input: dimnum= dimension

% fnum= function number

% xnum= initial iterate number

% tol= stoping tolarance

% maxit= maximum nuber of iteration

tic;

%%%%% default maxit, fev and tol, constant input %%%%%%%%%%%%%%%%%%%%%%%

if nargin<7

maxfev=2000;

end

if nargin<6

maxit=1000; % default max. iter

end

if nargin<5

tol=10^(-6); % default tolarance

end

%%%%%%%%%%%%% variable input %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<4

xlrange=[]; % excel range

end

if nargin<3

xnum=1; % default initial point

end

if nargin<2

dimnum=1; % default problem

end

if nargin<1

fnum=1; % default dimension

end

%%%%%%%%%%%%%%%%%%% defining dimension%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch dimnum

case 1

dim=1000;

case 2

dim=10000;

case 3

dim=50000;

case 4

dim=100000;

case 5

% case 6

% dim=10000;

% case 7

% dim=20000;

% case 8

% dim=50000;

% case 9

% dim=70000;

% case 10

% dim=100000;

otherwise

dim=dimnum; % for any other dimension

end

%%%%%%%%%%%%%%%%%% defining problems%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

problem=fnum;

switch problem

case 1

l=0; u=0;

f='cp2';

proj='Pj9';

case 2

l=0; u=+inf;

f='cp3';

proj='Pj';

case 3

l=-inf; u=0;

f='cp4';

proj='Pj';

case 4

l=0; u=+inf;

f='cp5';

proj='Pj';

case 5

l=0; u=0;

f='cp8';

proj='Pj2';

case 6

% l=-inf; u=0;

%f='cp4';

%proj='Pj';

% case 7

% l=-inf; u=0;

% f='cp7';

% proj='Pj';

% %l=0; u=0;

% case 8

l=0; u=0;

f='cp24';

proj='Pj9';

otherwise

f='fnum'; %for any other problem

end

%%%%%%%%%%%%%%%%%% defining initial points%%%%%%%%%%%%%%%%%%%%%%%

guess=xnum;

switch guess

case 1

%x0=rand(dim,1)

x0=(1./(1:dim))';

case 2

%x0=-0.5\*ones(dim,1);

x0=(1./(2.^(1:dim)))'; %1/2,1/4,1/8,...1/2^n;

case 3

x0=((1./(1:dim)).^2)'; %1,1/4,1/9,...1/n^2.

%x0=(1./(2.^(1:dim)))' %1/2,1/4,1/8,...1/2^n;

% x0=0.5\*ones(dim,1);

case 4

x0=(2./(1:dim))';

% x0=-10\*ones(dim,1);

case 5

x0=(1-(1./(1:dim)))'; %1-(1/n)

%x0=((1./(1:dim)).^2)'; %1,1/4,1/9,...1/n^2.

case 6

%x0=(1-(1./(1:dim)))'; %1-(1/n)

%x0=0.25\*ones(dim,1);

x0=((-1.^(1:dim))\*1./4)'; %-1/4,1/4,...,(1)^n(1/4)

case 7

x0=1.5\*ones(dim,1);

case 8

%x0=5\*ones(dim,1);

% case 9

% x0=(1./(1:dim))';

% case 10

% x0=10\*ones(dim,1);

% case 11

% x0=((dim-(1:dim))/dim)';

% case 12

% x0=rand(dim,1); % random numbers between 0 and 1

otherwise

x0=xnum; %for any other initial point

end

%Step 0 Initialization

ITER=0; %iteration

FEV=0; % function evaluation

bck=0; % backtracking counter

% line search parameters

%bita=1;

% gamma is for initial stepsize beta

%gam=1e-8;

% Step 1 stopping rule

F0=feval(f,x0); % evaluating F(x0);

FEV=FEV+1;

norm\_F0=sqrt(sum(F0.^2)); % norm of F(x0))

d0=-F0; % initial direction

%%%%%% Step 2 main loop%%%%%%%%%%%%%%%%%%%%%%%%%

while(ITER<=maxit && norm\_F0>tol)

ro=0.9; m=0; sig=0.0001; tau=0.01;

% p,q are for nonnegative parameter t

F0=feval(f,x0);

% dd1=F0'\*d0; % directional derivative (must be negative always!)

% %disp ('directional derivative', num2str(dd1))

% bita=dd1/(d0'\*(feval(f,x0+gam\*d0)-F0)/gam);

% if bita<=1e-6

% bita=1;

% end

gamma=1;

% Step 3: line search

while (((feval(f,x0+gamma\*(ro)^m\*d0))'\*d0)> -sig\*gamma\*(ro)^m\*(norm(feval(f,x0+gamma\*(ro)^m\*d0)))\*(norm(d0))^2 && m<=10)

m=m+1;

FEV=FEV+1;

end

if FEV>=maxfev

disp('maximum number of function evalution reached')

return;

end

% backtracking counter

if m

bck=bck+1;

end

alph=gamma\*(ro)^(m);

z=x0+alph\*d0;

Fz= feval(f,z); % computing f(z)

FEV=FEV+1;

if (feval(proj,(z),l,u)==z & norm(Fz)<tol)

x0=z;

F0=Fz;

norm\_F0= norm(F0);

disp('zk is in the convex set and its the solution at iteration number')

disp(num2str(ITER))

break

else

zetak=Fz'\*(x0-z)/(Fz'\*Fz); % computing zetak

P=feval(proj,(x0-zetak\*Fz),l,u); % projection on convex se

x=P;

F1=feval(f,x);

%s=x-x0;

s=z-x0; % Liu & Li choice of sk for DY-type projection method

y0=F1-F0;

% modified y using line search

A=(F1'\*y0)\*(F0'\*s)/max([norm(F1)^2\*(y0'\*s)+(F1'\*y0)\*(F0'\*s),tau\*norm(F1)\*norm(s)^2]);

B=norm(F1)^2\*(y0'\*s)/max([norm(F1)^2\*(y0'\*s)+(F1'\*y0)\*(F0'\*s),tau\*norm(F1)\*norm(s)^2]);

C=(y0'\*s)(F1'\*y0)\*(F0'\*s)/max([norm(y0)^2\*(norm(F1)^2\*(y0'\*s)+(F1'\*y0))\*(F0'\*s),tau\*norm(F1)\*norm(s)^4]);

L=A+B-C;

M=(F1'\*F1)\*d0-(F1'\*d0)\*F1;

N=max([tau\*norm(d0)\*norm(F1),(d0'\*y0)]);

L=M/N;

d1=-F1+(1-L)\*(F0'\*y0)/max([(y0'\*s),tau\*norm(s)^2])\*s+L\*norm(F1)^2/max([Fo\*s,norm(F1)\*norm(s)])\*s; % spectral Dai-Liao direction

end

x0=x;

F0=F1;

d0=d1;

norm\_F0=sqrt(sum(F0.^2));

ITER=ITER+1;

end

x0;

disp([num2str(ITER) ' / ' num2str(FEV) ' / ' num2str(bck) ' / ' num2str(toc) ' / ' num2str(norm\_F0) ])

disp(num2str(f))

disp(num2str(dim))

table1='STCGTABLE.xlsx';

T={ITER,FEV,toc,norm\_F0};

sheet=fnum;

xclRange=xlrange;

xlswrite(table1,T,sheet,xclRange);

% table1='dlcs.xlsx';

% T={num2str(ITER),num2str(FEV),num2str(toc),num2str(norm\_F0)};

% sheet=fnum;

% xlRange=xlrange;

% xlswrite(table1,T,sheet,xlRange);

%winopen(table1)

toc;

%MHCG IMAGECODE

%function signal(fnum,dimnum,xnum,xlrange,tol,maxit,maxfev)

% A Dai Liao type projection algorithm for convex constraints monotone equations

% with applications in compressive sensing

% A. B. Abubakar, H. Mohammad and P. Kumam March 30 2018

% Global convergence method

% call: dlcs(f,x0,tol,maxit)

% Input: dimnum= dimension

% fnum= function number

% xnum= initial iterate number

% tol= stoping tolarance

% maxit= maximum nuber of iteration

tic;

%%%%% default maxit, fev and tol, constant input %%%%%%%%%%%%%%%%%%%%%%%

if nargin<7

maxfev=2000;

end

if nargin<6

maxit=1000; % default max. iter

end

if nargin<5

tol=10^(-6); % default tolarance

end

%%%%%%%%%%%%% variable input %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<4

xlrange=[]; % excel range

end

if nargin<3

xnum=1; % default initial point

end

if nargin<2

dimnum=1; % default problem

end

if nargin<1

fnum=1; % default dimension

end

%%%%%%%%%%%%%%%%%%% defining dimension%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch dimnum

case 1

dim=1000;

case 2

dim=10000;

case 3

dim=50000;

case 4

dim=100000;

case 5

% dim=100000;

% case 6

% dim=10000;

% case 7

% dim=20000;

% case 8

% dim=50000;

% case 9

% dim=70000;

% case 10

% dim=100000;

otherwise

dim=dimnum; % for any other dimension

end

%%%%%%%%%%%%%%%%%% defining problems%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

problem=fnum;

switch problem

case 1

l=-inf; u=0;

f='cp1';

proj='Pj';

case 2

l=0; u=+inf;

f='cp2';

proj='Pj';

case 3

l=0; u=+inf;

f='cp5';

proj='Pj9';

case 4

l=0; u=+inf;

f='cp7';

proj='Pj';

% case 5

% l=0; u=0;

% f='cp8';

% proj='Pj2';

% case 6

% l=2; u=+inf;

% f='cp24';

% proj='Pj2';

otherwise

f='fnum'; %for any other problem

end

%%%%%%%%%%%%%%%%%% defining initial points%%%%%%%%%%%%%%%%%%%%%%%

guess=xnum;

switch guess

case 1

%x0=rand(dim,1)

%x0=(1./(1:dim))';

x0=1\*ones(dim,1);

case 2

x0=(1./(2.^(1:dim)))'; %1/2,1/4,1/8,...1/2^n.

%x0=0.5\*ones(dim,1);

case 3

% x0=-10\*ones(dim,1);

x0=((1./(1:dim)).^2)'; %1,1/4,1/9,...1/n^2.

case 4

x0=2\*ones(dim,1);

% x0=((1./(1:dim)).^2)'; %1,1/4,1/9,...1/n^2.

case 5

x0=(1-(1./(1:dim)))'; %1-(1/n)

%x0=0.25\*ones(dim,1);

case 6

x0=((-1.^(1:dim))\*1./4)'; %-1/4,1/4,...,(1)^n(1/4)

case 7

% x0=1.5\*ones(dim,1);

% case 8

% x0=5\*ones(dim,1);

% case 9

% x0=(1./(1:dim))';

% case 10

% x0=10\*ones(dim,1);

% case 11

% x0=((dim-(1:dim))/dim)';

% case 12

% x0=rand(dim,1); % random numbers between 0 and 1

otherwise

x0=xnum; %for any other initial point

end

%Step 0 Initialization

ITER=0; %iteration

FEV=0; % function evaluation

bck=0; % backtracking counter

% line search parameters

%bita=1;

% gamma is for initial stepsize beta

%gam=1e-8;

% Step 1 stopping rule

F0=feval(f,x0); % evaluating F(x0);

FEV=FEV+1;

norm\_F0=sqrt(sum(F0.^2)); % norm of F(x0))

d0=-F0; % initial

L=0.01;

%%%%%% Step 2 main loop%%%%%%%%%%%%%%%%%%%%%%%%%

while(ITER<=maxit && norm\_F0>tol)

ro=0.9; m=0; sig=0.0001; tau=0.01;

% p,q are for nonnegative parameter t

F0=feval(f,x0);

% dd1=F0'\*d0; % directional derivative (must be negative always!)

% %disp ('directional derivative ', num2str(dd1))

% bita=dd1/(d0'\*(feval(f,x0+gam\*d0)-F0)/gam);

% if bita<=1e-6

% bita=1;

% end

% gamma=1;

% Step 3: line search

while (((feval(f,x0+((ro)^m+((ro)^m)^2\*L)\*d0))'\*d0) > -sig\*(ro)^m+((ro)^m)^2\*L\*(norm(feval(f,x0+((ro)^m+((ro)^m)^2\*L)\*d0)))\*(norm(d0))^2 && m<=10)

m=m+1;

FEV=FEV+1;

end

if FEV>=maxfev

disp('maximum number of function0 evalution reached')

return;

end

% backtracking counter

if m

bck=bck+1;

end

alph=(ro)^(m);

mu=(alph+alph^2\*L);

z=x0+mu\*d0;

Fz= feval(f,z); % computing f(z)

FEV=FEV+1;

if (feval(proj,(z),l,u)==z & norm(Fz)<tol)

x0=z;

F0=Fz;

norm\_F0= norm(F0);

disp('zk is in the convex set and its the solution at iteration number')

disp(num2str(ITER))

break

else

zetak=Fz'\*(x0-z)/(Fz'\*Fz); % computing zetak

P=feval(proj,(x0-zetak\*Fz),l,u); % projection on convex se

x=P;

F1=feval(f,x);

y0=Fz-F0;

s=z-x0;

L1=((y0'\*y0)/(y0'\*s));

%tk=(F1'\*F1)\*(d0'\*d0)/(d0'\*y0)^2;

% computing the Descent Dai-Liao CG parameter

% betak=(F1'\*F1)/(d0'\*y0);

zt=((s'\*s)/(y0'\*s));

N=((y0'\*y0)-y0'\*s)\*y0'\*F1;

D=(y0'\*y0)\*y0'\*d0;

FR=N/D;

d1=-zt\*F1+FR\*d0; % spectral Dai-Liao direction

end

x0=x;

F0=F1;

d0=d1;

L=L1;

norm\_F0=sqrt(sum(F0.^2));

ITER=ITER+1;

end

x0;

disp([num2str(ITER) ' / ' num2str(FEV) ' / ' num2str(bck) ' / ' num2str(toc) ' / ' num2str(norm\_F0) ])

disp(num2str(f))

disp(num2str(dim))

table1='MHCG.xlsx';

T={ITER,FEV,toc,norm\_F0};

sheet=fnum;

xclRange=xlrange;

xlswrite(table1,T,sheet,xclRange);

% table1='dlcs.xlsx';

% T={num2str(ITER),num2str(FEV),num2str(toc),num2str(norm\_F0)};

% sheet=fnum;

% xlRange=xlrange;

% xlswrite(table1,T,sheet,xlRange);

%winopen(table1)

toc;