Exact Steepest Descent for the Model problem

```
% Demonstration program for the Steepest Descent Method
% The Test Problem:     f(x) =  b'x +x'A'x,    x*=-A^-1b.
%
%
% Define a random matrix
ndim = 80;              % Dimension of matrix
R    = randn(ndim);     % Random matrix with independent N(0,1)
%                         elements
%
npot = .3;              % Trick for the controlling the
%                         eigenvalue distribution
A  = (R'*R)^npot;       % A will be positive definite (A>0).
lamb = eig(A);          % Eigenvalues of A
kappa= max(lamb)/min(lamb); % The condition number
%
b    = rand(ndim,1);    % Generate a  random b-vector
xsol = -A\b;            % Compute The solution
%
Norm2 = sqrt(xsol'*xsol);     % 2-norm of the solution
NormA = sqrt(xsol'*A*xsol);   % A-norm of the solution
%
% Initialization of the SD iteration
x = zeros(size(b)); g = b;
Maxiterations = 100;
for loop = 1 : Maxiterations
   Ag   = A*g;                          % SD step
   alfa = (g'*g)./(g'*Ag);              % SD step
   x    = x - alfa*g;                   % SD step
   g    = g - alfa*Ag;                  % Alternative g = b+A*x;
   err2(loop) = sqrt((x-xsol)'*(x-xsol))/Norm2;    % Normalized error
   errA(loop) = sqrt((x-xsol)'*A*(x-xsol))/NormA;  % Normalized error
end;
%
% Display result
subplot(5,2,5);
plot(lamb,zeros(size(lamb)),'o');  % Plot of the Eigenvalues
axis([0 ceil(max(lamb)) -.5 .5]);
Tittel = ['Eigenvalues, matrix size =' num2str(ndim)];
title(Tittel);
%
subplot(1,2,2);
Iter = 1:Maxiterations;
KK = (kappa-1)/(kappa+1);
semilogy(Iter,err2,'b',Iter,errA,'r',Iter,KK.^Iter,'k');
legend( '2-norm' , 'A-norm', 'Error estimate' );
xlabel('Iteration number');
ylabel('Error')
Tittel = ['npot= ' num2str(npot) '  \kappa=',num2str(kappa)];
title(Tittel);
```

Line Search Steepest Descent for the Banana function

```
% Steepest descent demo for the Banana function
clf;
%
xlow  = -2;  % Define area (Large)
xhigh =  3;  % Define area
ylow  = -2;  % Define area
yhigh =  3;  % Define area
%
%xlow  =  0.8;  % Define area (small)
%xhigh = 1.2 ;  % Define area
%ylow  =  0.8;  % Define area
%yhigh =  1.2;  % Define area
%
% Make a contour plot of the banana function
%Define grid
[XG,YG] = meshgrid(xlow : .02 : xhigh, ylow : 0.02 : yhigh);
[B,dum,dum] = BananaGrad(XG,YG); % Compute function values
plot(1,1,'o') ;  % Plot solution point
axis([xlow xhigh ylow yhigh]);
axis('square')         % Same scale for x and y
grid                   % Grid lines
hold on                % Fix  plot and coordinate system
contour(XG,YG,B,25);   % Make contour plot
%
for ncases = 1:100       % Run many cases
  [x0,y0] = ginput(1);   % Pick a point
  for iter = 1:40        % The SD iteration
    X(iter) = x0;
    Y(iter) = y0;
    plot(X(1:iter),Y(1:iter)) % Plot the result while computing
    %
    % Prepares the SD-step: Uses fminbnd!
    % Compute function and gradient in current point:
    [b0,dbdx,dbdy] = BananaGrad(x0,y0);
    alphamax = 1;
    % Parameters for BananaLineSearch for BananaLineSearch
    P1 = x0 ; P2 = y0; P3 = -dbdx ; P4 = -dbdy;
    % Determines the "optimal" alpha. NB! there is more than one
    % local minimum!
    alphaopt = fminbnd('BananaLineSearch', 0  , alphamax ,…
    optimset('TolX',1e-8) ,P1,P2,P3,P4);
    x0 = P1 + alphaopt*P3;   % Update the current point
    y0 = P2 + alphaopt*P4;   % Update the current point
  end
end

function [b,dbdx,dbdy] = BananaGrad(x,y)
% Rosenbrock's Banana function and its gradient
a = 2;
b    = a*(y-x.*x).^2 + (1-x).^2 ;
dbdx = 4*a*x.*(x.*x-y) - 2*(1-x);
dbdy = 2*a*(y-x.*x);
```

```
function bval = BananaLineSearch(alpha,P1,P2,P3,P4);
x = P1 + alpha*P3;
y = P2 + alpha*P4;
[bval,dum1,dum2]=BananaGrad(x,y);
```

Demonstration of fminbnd

```
options = optimset('TolX',1.0e-15, 'Display','iter');
x = fminbnd (inline('exp(-x)+x/3'), 0, 6, options)
```

(For double precision numbers: Change the write statement in the fminbnd code).

Experiments with fminsearch

(not including the plot of the simplex)

```
clf
% Define area
xlow =  -2;
xhigh=  3;
ylow =  -2;
yhigh=  3;
% Make up contour plot
[XG,YG] = meshgrid(xlow : .05 : xhigh, ylow : 0.05 : yhigh);
[B,dum,dum] = BananaGrad(XG,YG) % Need only function value!
plot(1,1,'o') ;
axis([xlow xhigh ylow yhigh]);
grid;
hold on
contour(XG,YG,B,25,'k')
%
for k = 1:30  % Iteration
  [X0,Y0] = ginput(1);
  options = optimset('TolX',1.0e-15);
  options = optimset('MaxIter',70);
  options = optimset('Display','on');
  x = fminsearch('BananaFMS',[X0, Y0],options);
end;

function f = BananaFMS(x)
a= 10;% Original function = 100
f = a*(x(2)-x(1).^2).^2+(1-x(1)).^2;
```