Assignment 1 - Exercise 3

Table of Contents

Name: Stefano Gonçalves Simao

Date: 26/3/2021

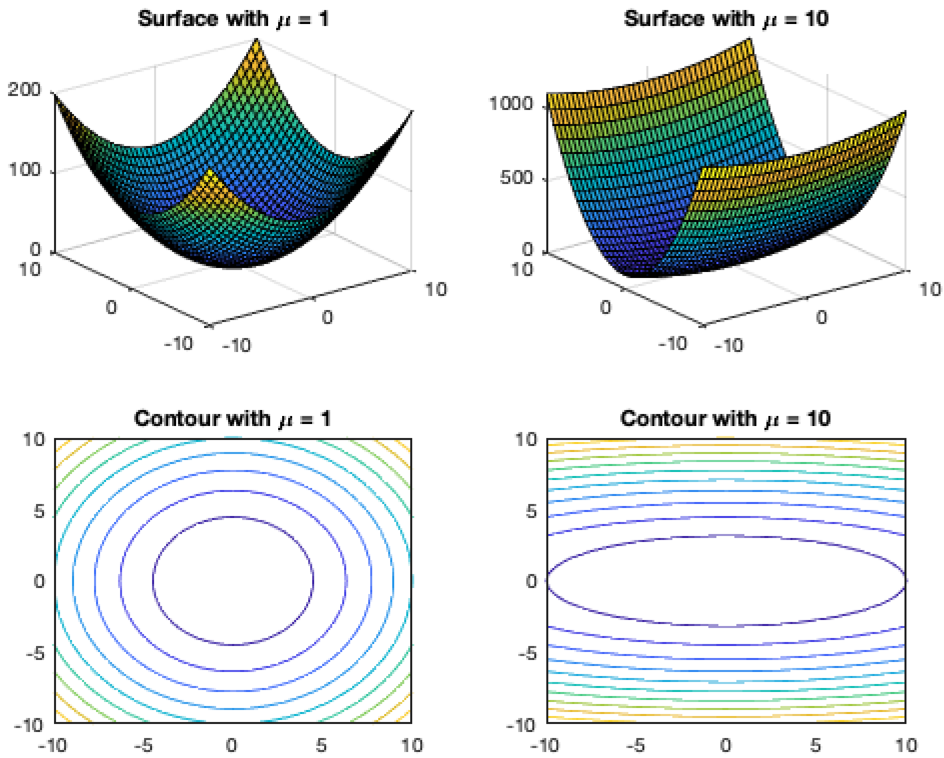
(2)

```
close all;
clc;

disp('(2)');
figure(1);
h1 = subplot (2,2,1);
mu = 1;
[x,y] = meshgrid(-10:0.5:10, -10:0.5:10);
f = x.^2 + mu * y.^2;
surf(x,y,f)
title('Surface with \mu = 1');

h2 = subplot (2,2,2);
mu = 10;
[x,y] = meshgrid(-10:0.5:10, -10:0.5:10);
f = x.^2 + mu * y.^2;
surf(x,y,f)
title('Surface with \mu = 10');

h3 = subplot (2,2,3);
mu = 1;
[x,y] = meshgrid(-10:0.5:10, -10:0.5:10);
f = x.^2 + mu * y.^2;
contour(x,y,f)
title('Contour with \mu = 1');

h4 = subplot (2,2,4);
mu = 10;
[x,y] = meshgrid(-10:0.5:10, -10:0.5:10);
f = x.^2 + mu * y.^2;
contour(x,y,f)

title('Contour with \mu = 10');

disp('------------------------------------------------------------------')
```

```
disp('The starting points in the function with μ = 10 are the ones
 that could be problematic.')
disp("It's much difficult to find a good starting point and it should
 be where the contour lines")
disp("meet the axes as the gradient of a function is always
 perpendicular to the")
disp("contour lines (a good place would be with any y but around x =
 0). Here the convergence will")
disp("be slower as it will zigzag much more. This is beacause the
 contour lines are ellipses whose")
disp('axes lie along the orthogonal lines of A, and with this
 difference in magnitude, as said,')
disp('the steepest descent will oscillate.')
disp("In the other case it's much easier for the algorithm to converge
 as every starting point has")
disp("fast convergence. It should be able to converge in 1
 iteration.")
disp('-------------------------------------------------------------------')
disp(' ')
```

(2)

```
-------------------------------------------------------------------
The starting points in the function with μ = 10 are the ones that
 could be problematic.
It's much difficult to find a good starting point and it should be
 where the contour lines
meet the axes as the gradient of a function is always perpendicular to
 the
contour lines (a good place would be with any y but around x = 0).
 Here the convergence will
be slower as it will zigzag much more. This is beacause the contour
 lines are ellipses whose
axes lie along the orthogonal lines of A, and with this difference in
 magnitude, as said,
the steepest descent will oscillate.
In the other case it's much easier for the algorithm to converge as
 every starting point has
fast convergence. It should be able to converge in 1 iteration.
-------------------------------------------------------------------
```



(4 & 5)

```
disp('(4 & 5)')
disp('-------------------------------------------------------------------')
disp('Not knowing exacly what is required in the assignment, I decided
 to implement both Gradient descent')
disp('and Conjugate Gradient methods. Setting beta = 0 in the CD
 algorithm would make x computed by the')
disp('gradient descent method. One thing to note here is that in the
 plots that are functions of')
disp('the iterations, the iteration starts at number 2, the iteration
 No.1 is the status before ')
disp('the start of the loop.')
disp('-------------------------------------------------------------------')
%Gradient descent first
figure('Name', 'Gradient descent');
v = tiledlayout(2,3);
title(v,'Gradient descent')
[x1, r1, F1]=GD(1, [10;0]);
[x2, r2, F2]=GD(1, [0;10]);
[x3, r3, F3]=GD(1, [10;10]);

[x1, r4, F4]=GD(10, [10;0]);
[x2, r5, F5]=GD(10, [0;10]);
[x3, r6, F6]=GD(10, [10;10]);
```

```
%Gradient norm
figure('Name', 'Gradient descent - log10 norm of the Gradient');
z = tiledlayout(2,3);
nexttile;
semilogy(r1, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 1 and x0 = (10,0)');
title(s)

nexttile;
semilogy(r2, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 1 and x0 = (0,10)');
title(s)

nexttile;
semilogy(r3, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 1 and x0 = (10,10)');
title(s)

nexttile;
semilogy(r4, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 10 and x0 = (10,0)');
title(s)

nexttile;
semilogy(r5, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 10 and x0 = (0,10)');
title(s)

nexttile;
semilogy(r6, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 10 and x0 = (10,10)');
title(s)
title(z,'Gradient descent - Energy function')

%Enery function value
figure('Name', 'Gradient descent - Energy function');
j = tiledlayout(2,3);
nexttile;
plot(F1, '-go');
xlabel('Iteration');
ylabel('Function value');
```

```
s = sprintf('μ = 1 and x0 = (10,0)');
title(s)

nexttile;
plot(F2, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 1 and x0 = (0,10)');
title(s)

nexttile;
plot(F3, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 1 and x0 = (10,10)');
title(s)

nexttile;
plot(F4, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 10 and x0 = (10,0)');
title(s)

nexttile;
plot(F5, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 10 and x0 = (0,10)');
title(s)

nexttile;
plot(F6, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 10 and x0 = (10,10)');
title(s)
title(j,'Gradient descent - Energy function')

disp('------------------------------------------------------------------')
%Gonjugate gradient second
figure('Name', 'Conjugate gradient');
t = tiledlayout(2,3);
title(t,'Conjugate gradient')
[x1, r1, F1]=CG(1, [10;0]);
[x2, r2, F2]=CG(1, [0;10]);
[x3, r3, F3]=CG(1, [10;10]);

[x1, r4, F4]=CG(10, [10;0]);
[x2, r5, F5]=CG(10, [0;10]);
[x3, r6, F6]=CG(10, [10;10]);

%Gradient norm
figure('Name', 'Conjugate gradient - log10 norm of the Gradient');
```

```
g = tiledlayout(2,3);
nexttile;
semilogy(r1, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 1 and x0 = (10,0)');
title(s)

nexttile;
semilogy(r2, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 1 and x0 = (0,10)');
title(s)

nexttile;
semilogy(r3, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 1 and x0 = (10,10)');
title(s)

nexttile;
semilogy(r4, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 10 and x0 = (10,0)');
title(s)

nexttile;
semilogy(r5, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 10 and x0 = (0,10)');
title(s)

nexttile;
semilogy(r6, '-r+');
xlabel('Iteration');
ylabel('Gradient norm');
s = sprintf('μ = 10 and x0 = (10,10)');
title(s)
title(g,'Conjugate gradient - log10 norm of the Gradient')

%Enery function value
figure('Name', 'Conjugate gradient - Energy function');
u = tiledlayout(2,3);
nexttile;
plot(F1, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 1 and x0 = (10,0)');
title(s)
```

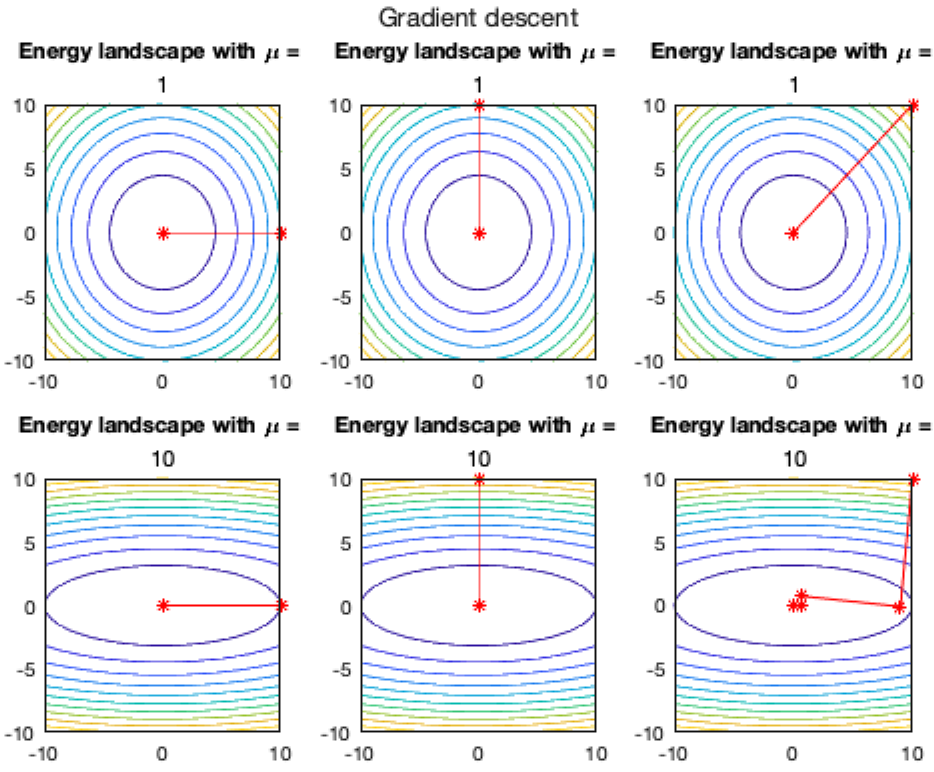
```
nexttile;
plot(F2, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 1 and x0 = (0,10)');
title(s)

nexttile;
plot(F3, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 1 and x0 = (10,10)');
title(s)

nexttile;
plot(F4, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 10 and x0 = (10,0)');
title(s)

nexttile;
plot(F5, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 10 and x0 = (0,10)');
title(s)

nexttile;
plot(F6, '-go');
xlabel('Iteration');
ylabel('Function value');
s = sprintf('μ = 10 and x0 = (10,10)');
title(s)
title(u,'Conjugate gradient - Energy function')

disp('-----------------------------------------------------------------')
disp("We can see that with μ = 1 there are no problems, each algorithm
 finds the solution in");
disp("one iteration as the negative gradient points to the solution.
 With μ = 10 it's more interesting");
disp("as for (x0,y0) = (10,0) and (0,10) the starting position is a
 good choice for both");
disp("algorithms. (10,10) is trickier here as in GD it zigzags more as
 it is not a good ");
disp("starting point. With CG we are sure to come to the solution in n
 iterations, in this");
disp("case we have n = 2. So it is anyway more effective even with a
 starting point that is not good.");
disp("In the case of μ = 10 and starting point (10,10), both
 algorithms don't provide an exact");
disp("solution, but one that is very close to (0,0). It is inside the
 tol given.");
```

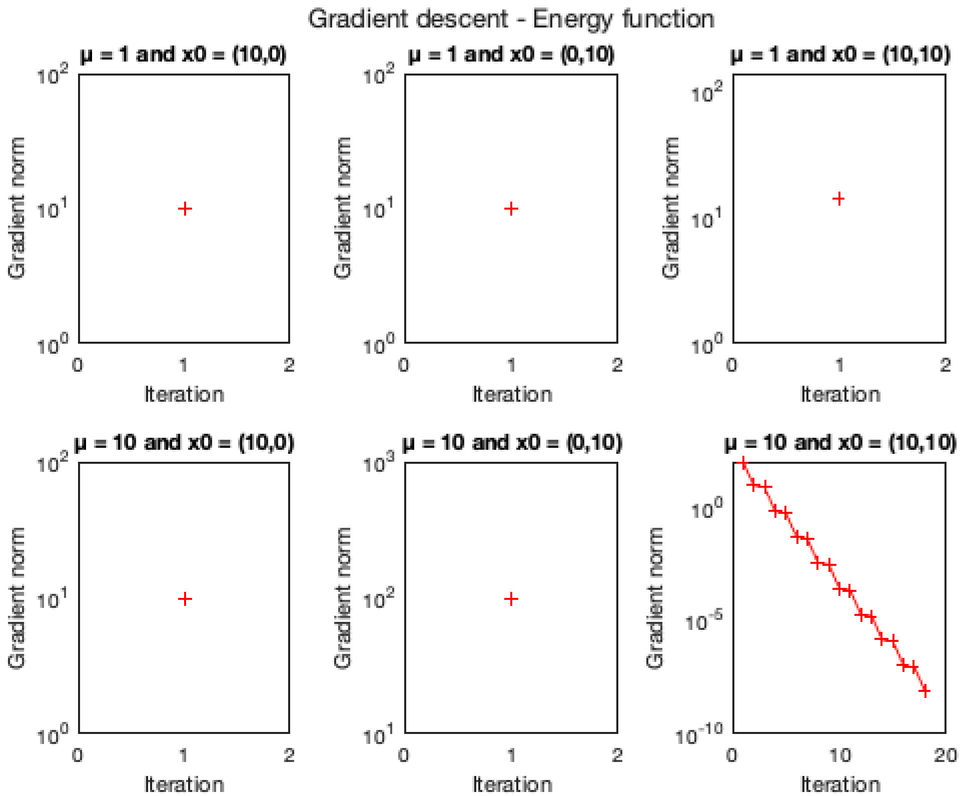
```
disp("Another thing to note is that in the log10 of the norm of the
 gradient ");
disp("the zero is not represented as log(0) = Inf for Matlab and it
 doesn't plot it.");
disp('-------------------------------------------------------------------')
```

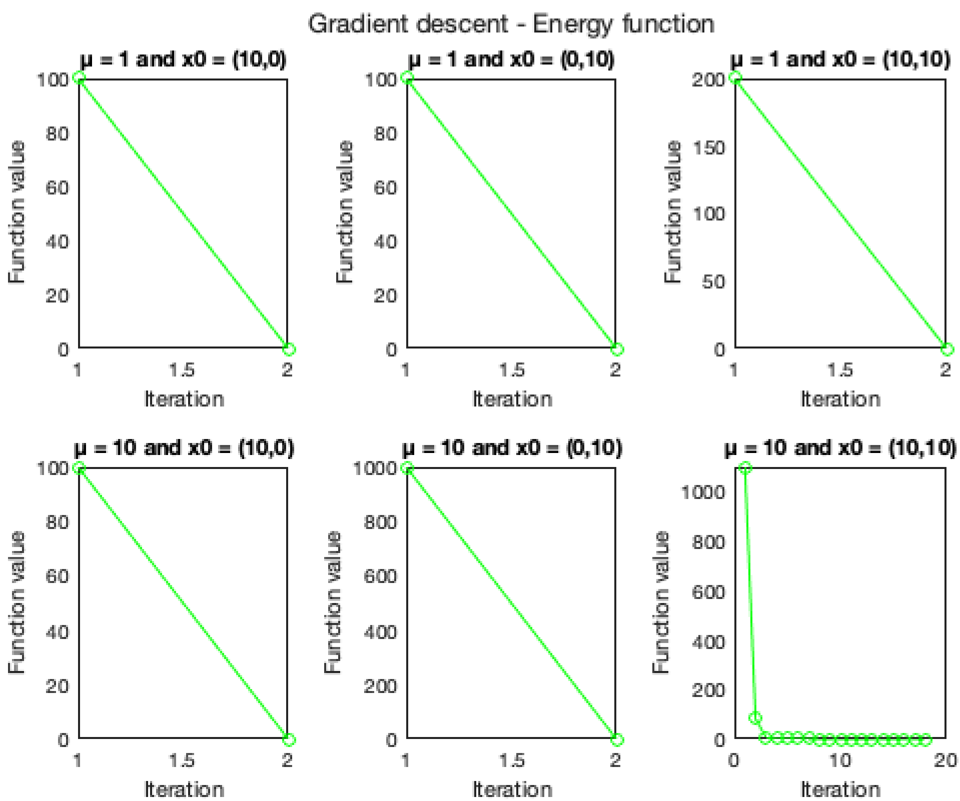
```
(4 & 5)
-------------------------------------------------------------------
Not knowing exacly what is required in the assignment, I decided to
 implement both Gradient descent
and Conjugate Gradient methods. Setting beta = 0 in the CD algorithm
 would make x computed by the
gradient descent method. One thing to note here is that in the plots
 that are functions of
the iterations, the iteration starts at number 2, the iteration No.1
 is the status before
the start of the loop.
-------------------------------------------------------------------
GD
 μ = 1: and starting point (10,0)
x =
     0
     0

---------------------
GD
 μ = 1: and starting point (0,10)
x =
     0
     0

---------------------
GD
 μ = 1: and starting point (10,10)
x =
     0
     0

---------------------
GD
 μ = 10: and starting point (10,0)
x =
     0
     0

---------------------
GD
 μ = 10: and starting point (0,10)
x =
     0
     0

---------------------
GD
```

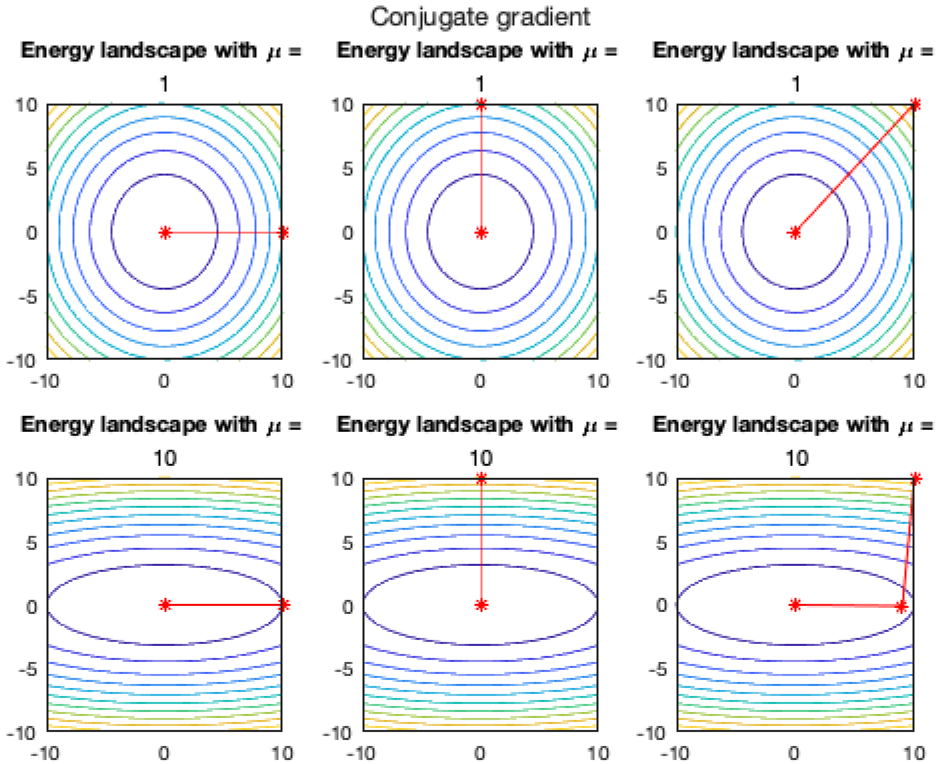
```
 μ = 10: and starting point (10,10)
x =
   1.0e-08 *

    0.7710
   -0.0077

---------------------
-------------------------------------------------------------------
CG
 μ = 1: and starting point (10,0)
x =
     0
     0

--------------------------------------------
CG
 μ = 1: and starting point (0,10)
x =
     0
     0

--------------------------------------------
CG
 μ = 1: and starting point (10,10)
x =
     0
     0

--------------------------------------------
CG
 μ = 10: and starting point (10,0)
x =
     0
     0

--------------------------------------------
CG
 μ = 10: and starting point (0,10)
x =
     0
     0

--------------------------------------------
CG
 μ = 10: and starting point (10,10)
x =
   1.0e-14 *

   -0.1776
    0.2012

--------------------------------------------
-------------------------------------------------------------------
```

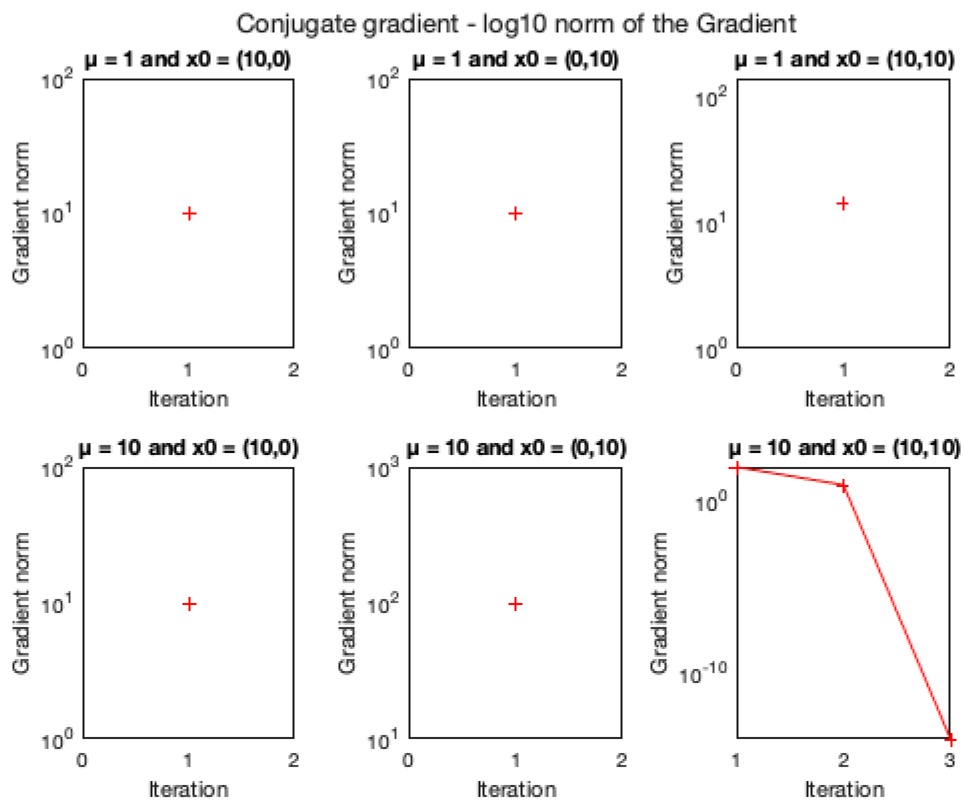
```
We can see that with μ = 1 there are no problems, each algorithm finds
 the solution in
one iteration as the negative gradient points to the solution. With μ
 = 10 it's more interesting
as for (x0,y0) = (10,0) and (0,10) the starting position is a good
 choice for both
algorithms. (10,10) is trickier here as in GD it zigzags more as it is
 not a good
starting point. With CG we are sure to come to the solution in n
 iterations, in this
case we have n = 2. So it is anyway more effective even with a
 starting point that is not good.
In the case of μ = 10 and starting point (10,10), both algorithms
 don't provide an exact
solution, but one that is very close to (0,0). It is inside the tol
 given.
Another thing to note is that in the log10 of the norm of the
 gradient
the zero is not represented as log(0) = Inf for Matlab and it doesn't
 plot it.
-------------------------------------------------------------------
```

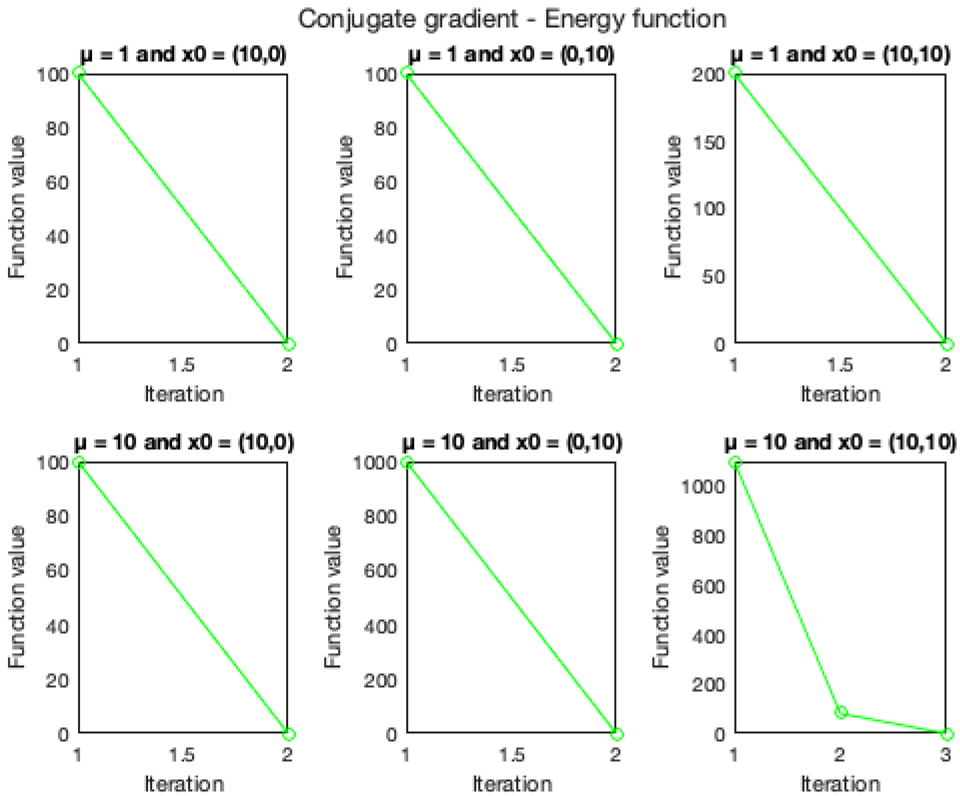



Gradient descent - Energy function
μ = 1 and x0 = (10,0)
μ = 1 and x0 = (0,10)
μ = 1 and x0 = (10,10)
μ = 10 and x0 = (10,0)
μ = 10 and x0 = (0,10)
μ = 10 and x0 = (10,10)
Gradient norm
Iteration


Gradient descent - Energy function
μ = 1 and x0 = (10,0)
μ = 1 and x0 = (0,10)
μ = 1 and x0 = (10,10)
μ = 10 and x0 = (10,0)
μ = 10 and x0 = (0,10)
μ = 10 and x0 = (10,10)
Function value
Iteration


Conjugate gradient
Energy landscape with μ = 1
Energy landscape with μ = 1
Energy landscape with μ = 1
Energy landscape with μ = 10
Energy landscape with μ = 10
Energy landscape with μ = 10
10
5
0
-5
-10
-10
0
10


Conjugate gradient - log10 norm of the Gradient
μ = 1 and x0 = (10,0)
μ = 1 and x0 = (0,10)
μ = 1 and x0 = (10,10)
μ = 10 and x0 = (10,0)
μ = 10 and x0 = (0,10)
μ = 10 and x0 = (10,10)
Gradient norm
Iteration



"Gradient Descent"

```
function [x, rvec, F]=GD(mu,x0)
%Gradient descent method

%Initialization
A = [1 0; 0 mu];
b = [0; 0];
x = x0;
rvec = [];
Xarr = [];
Yarr = [];
F = [];
max_itr = 100;
tol = 10^(-8);

nexttile;
formatSpec = 'GD\n μ = %d: and starting point (%d,%d)\n';
fprintf(formatSpec, mu, x0(1,1), x0(2,1));
[X,Y] = meshgrid(-10:0.5:10, -10:0.5:10);
f = X.^2 + mu * Y.^2;
contour(X,Y,f)
title('Energy landscape with \mu = ', num2str(mu));
hold on;
```

```
r = b - A * x0;
d = r;
p_old = dot(r,r);
i = 1;
%The first iteration is actually the status before it starts.
 Iteration 1
%starts at position 2
rvec(i) = norm(r);
F(i) = x0(1,1)^2 + mu * x0(2,1)^2;

Xarr(1) = x0(1,1);
Yarr(1) = x0(2,1);

while (i < max_itr && norm(r) >= tol)
    s = A * d;
    alpha = p_old / dot(d, s);
    x = x + alpha * d;
    r = r - alpha * s;
    p_new = dot(r,r);
    beta = 0;
    d = r + beta * d;
    p_old = p_new;
    i = i + 1;
    Xarr(end + 1) = x(1,1);
    Yarr(end + 1) = x(2,1);
    rvec(i) = norm(r);
    F(i) = x(1,1)^2 + mu * x(2,1)^2;
end
plot (Xarr,Yarr, '-r*');
disp('x =');
disp(x);
disp('----------------------');
end
```

"Conjugate gradient"

```
function [x, rvec, F]=CG(mu,x0)
%Conjugate Gradient method

%Initialization
A = [1 0; 0 mu];
b = [0; 0];
x = x0;
rvec = [];
Xarr = [];
Yarr = [];
F = [];
max_itr = 100;
tol = 10^(-8);
```

```
nexttile;
formatSpec = 'CG\n μ = %d: and starting point (%d,%d)\n';
fprintf(formatSpec, mu, x0(1,1), x0(2,1));
[X,Y] = meshgrid(-10:0.5:10, -10:0.5:10);
f = X.^2 + mu * Y.^2;
contour(X,Y,f)
title('Energy landscape with \mu = ', num2str(mu));
hold on;

r = b - A * x0;
d = r;
p_old = dot(r,r);
i = 1;
%The first iteration is actually the status before it starts.
 Iteration 1
%starts at position 2
rvec(i) = norm(r);
F(i) = x0(1,1)^2 + mu * x0(2,1)^2;

Xarr(1) = x0(1,1);
Yarr(1) = x0(2,1);

while (i < max_itr && norm(r) > tol)
    s = A * d;
    alpha = p_old / dot(d, s);
    x = x + alpha * d;
    r = r - alpha * s;
    p_new = dot(r,r);
    beta = p_new/p_old;
    d = r + beta * d;
    p_old = p_new;
    i = i + 1;
    Xarr(end + 1) = x(1,1);
    Yarr(end + 1) = x(2,1);
    rvec(i) = norm(r);

    F(i) = x(1,1)^2 + mu * x(2,1)^2;
end
plot (Xarr,Yarr, '-r*');
disp('x =');
disp(x);
disp('---------------------------------------------');
end
```

Published with MATLAB® R2020b