

COMPM012 - Coursework 1

Esben A. Sørig

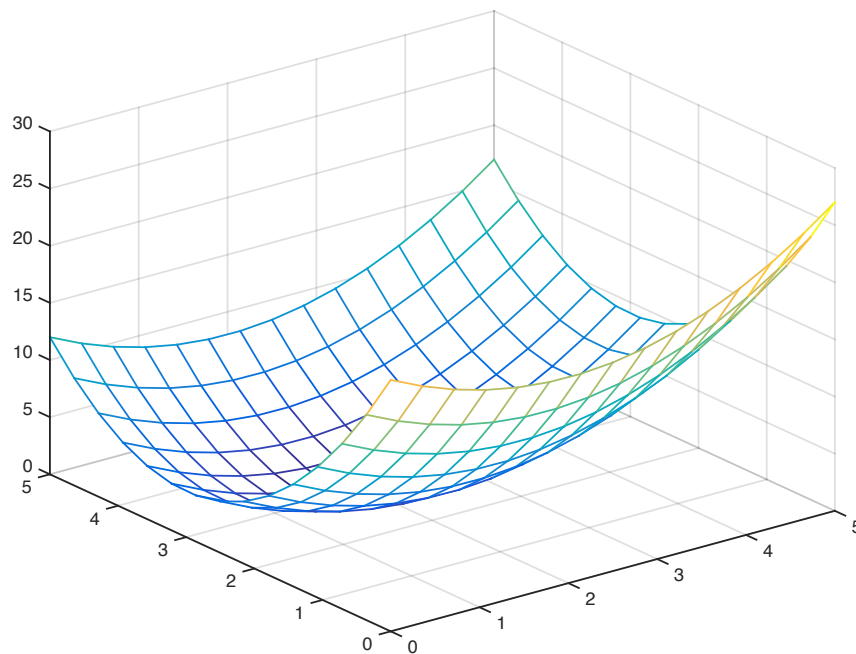
November 27, 2014

1 Gradient Descent Visualisation

a) To produce the plot the following commands are used:

```
>> [X,Y] = meshgrid(linspace(0,5,15),linspace(0,5,15));  
>> mesh(X,Y,fcarg(X,Y));
```

Which gives us the plot



b) i) The modified graddesc function becomes:

```

function [soln , X, Y, Z] = graddesc(f, g, i,e, t)
    gi = feval(g,i) ;
    X = [];
    Y = [];
    Z = [];
    while(norm(gi)>t) % crude termination condition
        i = i - e .* feval(g,i) ;
        gi = feval(g,i);
        X = [X i(1)];
        Y = [Y i(2)];
        Z = [Z fc(i)];
    end
    soln = i ;
end

```

ii) We produce the plot with the commands

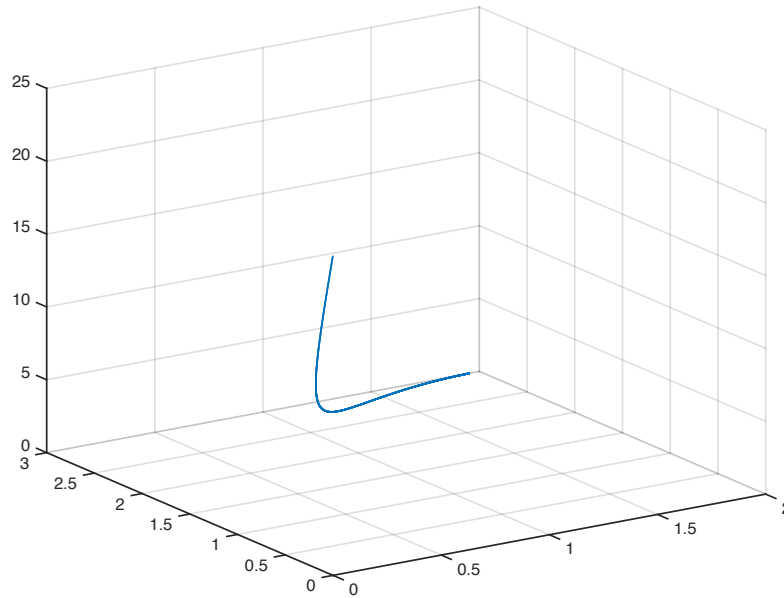
```

% Gradient descent
[result , X, Y, Z] = graddesc('fc','dfc',[0,0],0.001,0.1);

% Plot descent path
hold on
plot3(X, Y, Z)
grid on
hold off

```

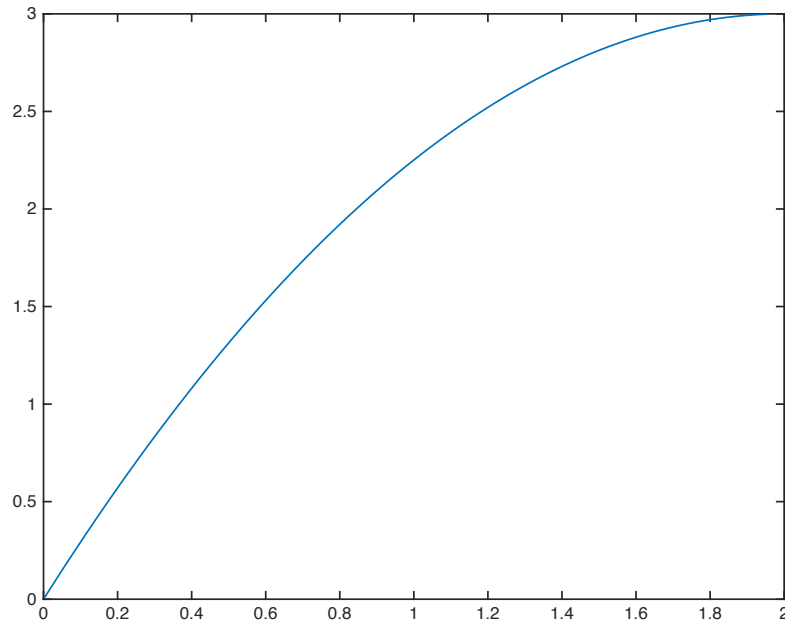
The obtained plot looks like



- iii) We can plot the projection to the XY-plane by simply plotting the X, Y values and ignore the Z values. We do this with the following commands

```
% Plot projection to XY plane  
plot(X, Y)
```

Which produces the following plot



2 Gradient Descent for Linear Regression

a) My implementation of gradient descent is

```
function [sol , guesses] = mygraddesc(A, b, guess , step , tol)
    mse = @(w) (A*w - b).' * (A*w - b);
    dmsedw = @(w) (-2*A.' * b) + (2*A.' * A * w);
    guesses = [guess; mse(guess)]; % Used for visualisation

    while norm(dmsedw(guess)) > tol
        guess = guess - dmsedw(guess)*step;
        guesses = [guesses [guess; mse(guess)]];
    end

    sol = guess;
end
```

b) We can find the solution to the system using the gradient descent implementation by putting the coefficients for x_1 and x_2 in a matrix A and the right-hand-sides of the equations in a vector b . Passing A and

b as arguments to the gradient descent function along with an initial guess, step size, and tolerance gives us the solution to the system. In Matlab we do:

```
A = [1 -1; 1 1; 1 2]
b = [1; 1; 3];
guess = [0; 0];
```

```
mygraddesc(A, b, guess, 0.01, 0.0001)
```

Which returns

```
ans =
    1.2857
    0.5714
```

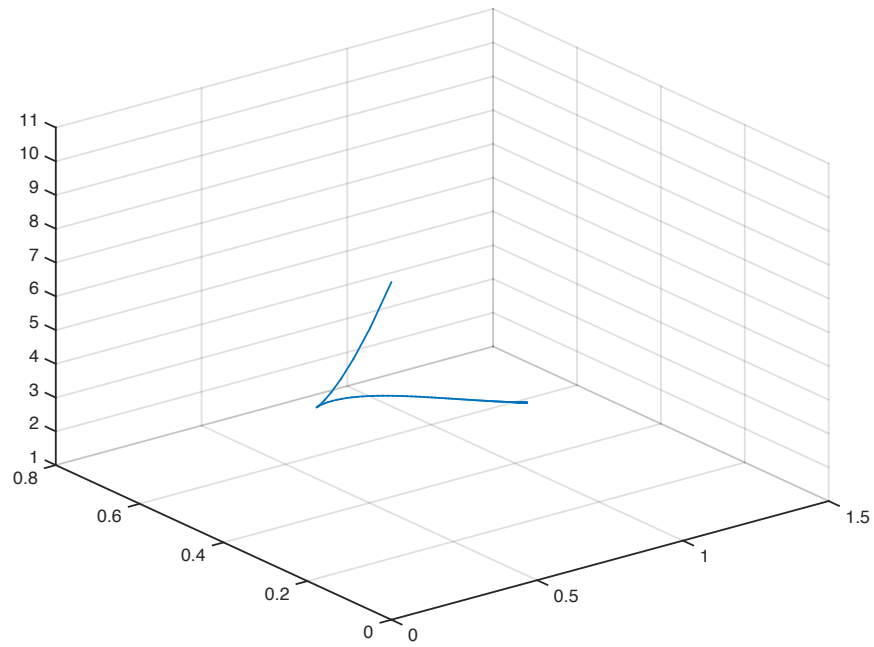
So $x_1 = 1.2857$ and $x_2 = 0.5714$ is the least squares solution to the system.

- c) My gradient descent implementation lets us retrieve the guesses for each iteration of the descent. We plot this as follows

```
[~, guesses] = mygraddesc(A, b, guess, 0.01, 0.001);
```

```
plot3(guesses(1,:), guesses(2,:), guesses(3,:));
hold on
grid on
hold off
```

Which produces the plot



3 Convergence of Gradient Descent in a Single Variable

- a)
- b)
- c)
- d)