Lab 12: SVD

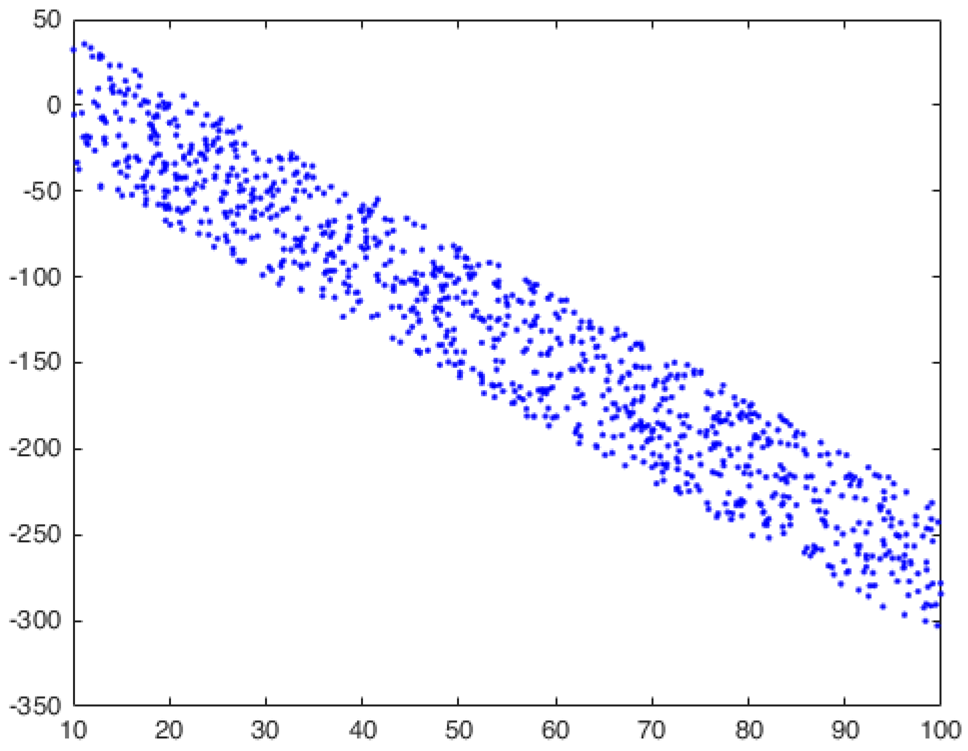
Two ways to fit point cloud to a line

1 - take SVD of our matrix Ax=0 (we need to put b in to A matrix), take the smallest V vector it will be a solution.

2 - Classical least square solution via pseudo inverse

2nd approach is more accurate, it can be seen by error comparison

```
% Generate some points around a line
intercept = -10; slope = -3;
npts = 1000; noise = 80;
xs = 10 + rand(npts, 1) * 90;
ys = slope * xs + intercept + rand(npts, 1) * noise;
% xs = [1;2;3]
% ys = [1;2;1]
% npts = 3;
% Plot the randomly generated points
figure; plot(xs, ys, 'b.', 'MarkerSize', 5)
```



```
% Fit these points to a line - 1st approach
A = [xs, ys, -1 * ones(npts, 1)];
[U, S, V] = svd(A);
fit = V(:, end-1)
```

```
fit =
   -0.9326
   -0.3590
    0.0362
```

```
% Get the coefficients a, b, c in ax + by + c = 0
a = fit(1); b = fit(2); c = fit(3);

% Compute slope m and intercept i for y = mx + i
slope_est = -a/b;
intercept_est = c/b;

% Plot fitted line on top of old data
ys_est = slope_est * xs + intercept_est;
figure; plot(xs, ys, 'b.', 'MarkerSize', 5);
hold on; plot(xs, ys_est, 'r-')
% Error
sum_err_line = sum((ys_est-ys).^2)
```

```
sum_err_line = 7.0883e+05
```

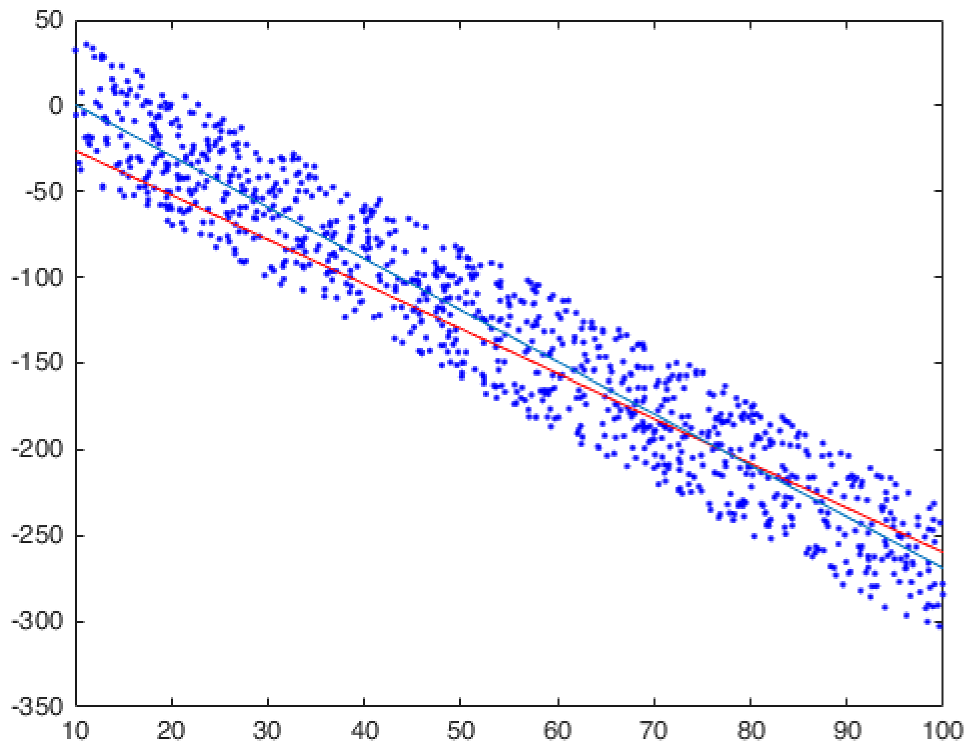
```
% Fit these points to a line - 2nd approach
fit1 = pinv([A(:,1) 1 * ones(npts, 1)])*A(:,2)
```

```
fit1 =
   -2.9980
   30.6925
```

```
k = fit1(1); b = fit1(2);
slope_est1 = k;
intercept_est1 = b;
ys_est1 = slope_est1 * xs + intercept_est1;
% Error
sum_err_line1 = sum((ys_est1-ys).^2)
```

```
sum_err_line1 = 5.2035e+05
```

```
% Blue one - 2nd approach, Red one - 1st
hold on; plot(xs, ys_est1)
```


50
0
-50
-100
-150
-200
-250
-300
-350
10
20
30
40
50
60
70
80
90
100