Exercise1: Estimating velocity motion model of a mobile robot through linear regression

Run: par = Exercise1(k)
Functions for parameters a estimation using linear regression: estimate_xy.m, estimate_theta.m
Functions for prediction using the estimated parameters: predict_theta.m, predict_xy.m
Functions for getting the optimal polynomial parameter estimation: findopt.m
Result:
for k = 2, p1 =5, p2 = 3:

a1	a2	a3
optt_a1 = 0.0022 0.9217 0.0066 -0.0016 -0.0010 0.0025 0.0023 -0.0000 -0.0130 0.0001 0.0000 -0.0045 -0.0000 0.0000 0.0026 -0.0000	optt_a2 = -0.0027 -0.0014 -0.0115 0.4730 0.0002 -0.0083 0.0001 0.0000 0.0164 -0.0010 -0.0000 0.0043 -0.0000 -0.0000 -0.0038 0.0000	optt_a3 = -0.0006 -0.0002 0.9997 0.0008 0.0001 0.0018 -0.0001 -0.0000 -0.0006 -0.0000

for k = 4, p1 =4, p2 = 1:

a1	a2	a3
optt_a1 = 0.0025 0.9198 -0.0029 -0.0007 -0.0010 0.0014 0.0025 0.0001 -0.0003 0.0001 0.0000 -0.0043 -0.0000	optt_a2 = -0.0043 -0.0010 0.0014 0.4680 0.0006 -0.0025 -0.0010 0.0000 -0.0017 -0.0007 -0.0000 0.0035 0.0000	optt_a3 = 0.0008 -0.0003 0.9987 0.0003

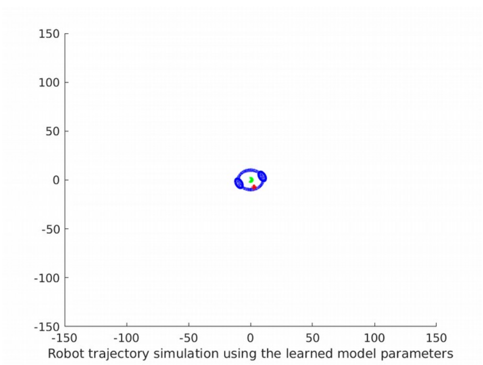
For different k, optimal polynomial is different:

K	1	2	4	5	8	10
Optimal p1	Warning	5	4	4	4	4
Optimal p2	Warning	3	1	1	2	1

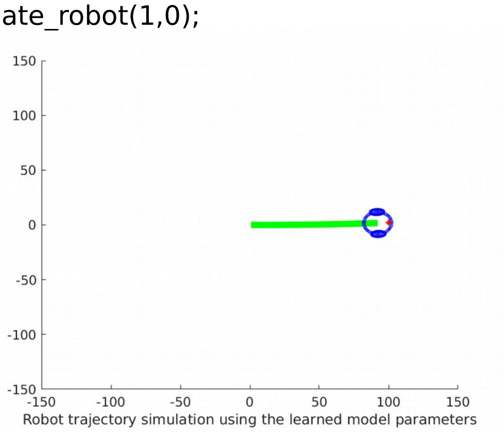
If I set maxima polynomial more than 6, the result of optimal polynomials won't change.

Plots: for k = 5, p1= 4, p2 = 1

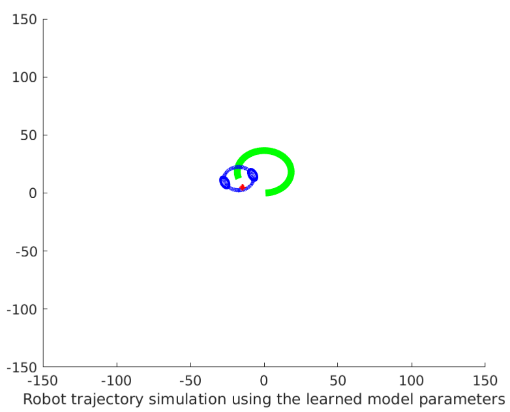
Simulate_robot(0,0.05);



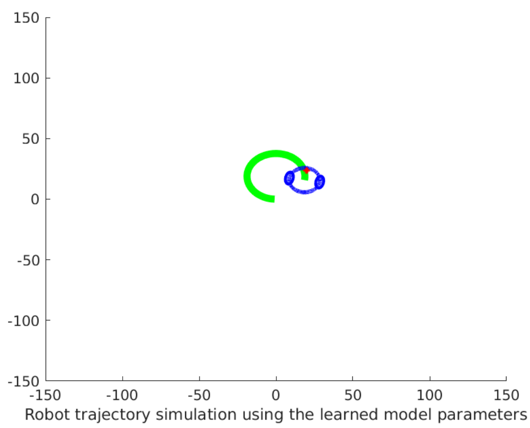
Simulate_robot(1,0);



Simulate_robot(1,0.05);



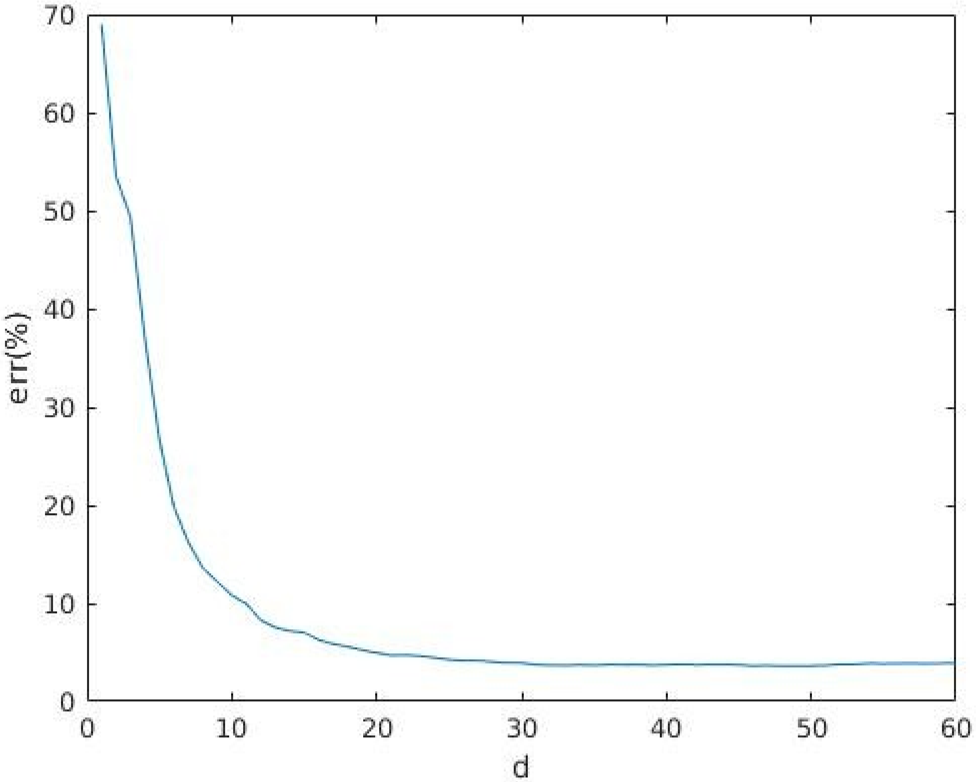
Simulate_robot(-1,-0.05)



From the Simulation, it is seen that the estimated robot trajectory varies if the staring velocity and angular velocity are different. This is not surprising since the estimated trajectory are added up to the starting position, once it changes, the robot position changes too.

Exercise 2: Handwritten digits classification using Baysian classifier

Run: [min_val,min_ind] = Exercise2(60)
Result:
1. optimal d = 48 with the optimal error rate: err = 3.62%
2. Plot of classification error err and dimensions d. From the plot, it is obvious that the error decreases when the target dimension of PCA increases. The minima error is minima by dimension d = 48, and it converges afterwards.

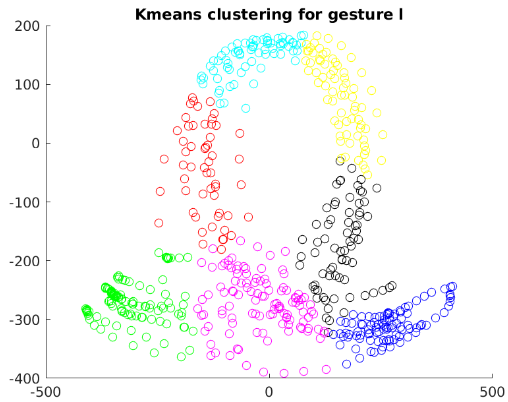
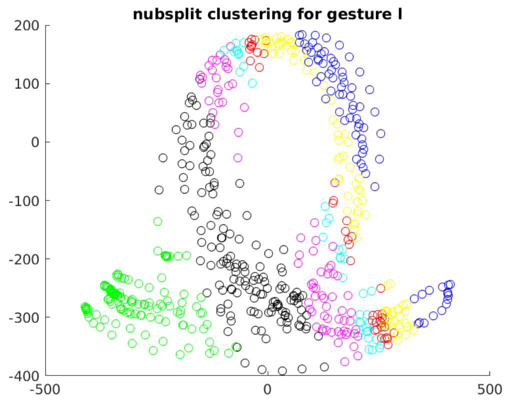
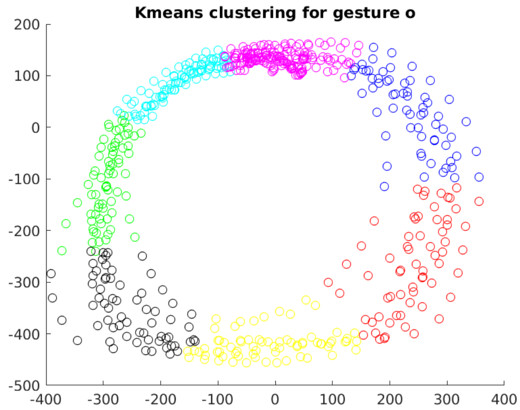
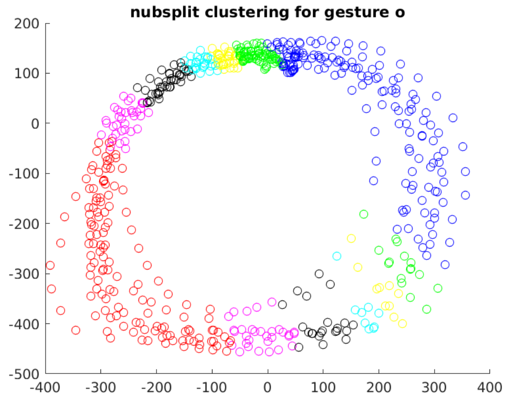
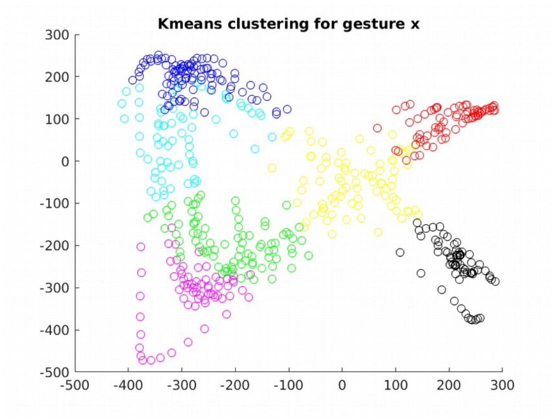
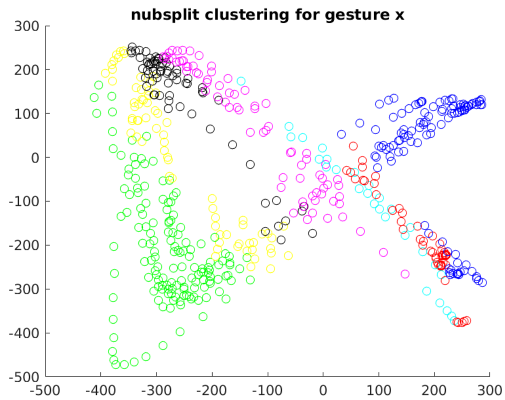


3. confusion matrix for d = 48:

digit	0	1	2	3	4	5	6	7	8	9
0	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
1	0.00	0.97	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00
2	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.02	0.00
3	0.00	0.00	0.01	0.96	0.00	0.00	0.00	0.00	0.02	0.00
4	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.01
5	0.00	0.00	0.00	0.02	0.00	0.96	0.00	0.00	0.01	0.00
6	0.01	0.00	0.00	0.00	0.00	0.01	0.96	0.00	0.01	0.00
7	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.93	0.01	0.02
8	0.00	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.97	0.01
9	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.01	0.01	0.94

Exercise3: Human motion clustering
Run: Exercise3_kmeans(), Exercise_nubs()
Function for kmeans cluster: kmeansclus.m
Function for: nubsplit.m
plot the labels for data: plotclus.m

	K-means Elapsed Time: 0.187s	Non-Uniform Binary Split Elapsed Time: 0.0446s

Gesture L		
Gesture O		
Gesture X		

From the plot, we can see that the performance of two algorithms are very different:

– K-means has better distinguished clusters than non-uniform binary classification

– but binary split is much more fast than k-means