Guidelines – Read Carefully! Please check each problem for problem-specific instructions. For the submission of this machine problem assignment, you should create a top-level folder named with your NETID, (e.g., XYZ007 for a single person group and XYZ007-ABC999 for a two-person group). Then, for each machine problem in a given assignment, you should create a sub-folder for that problem named with the problem number. For example, for this MP set, which has two problems, you should have the following folder structure if you are a single person group:

XYZ007
XYZ007/01
XYZ007/02

That is, you should have folders "01" and "02" under the folder "XYZ007". The files for each problem should go to the corresponding sub-folder. When you are ready to submit, remove any extra files (e.g., some python interpreter will create .pyc files) that are not required and zip the entire folder. The zip file should also be named with your NETID as XYZ007.zip or XYZ007-ABC999.zip. For this particular assignment, the folder structure is already created for you in the accompanied zip file; you just need to rename the top folder as instructed.

You are required to write your program adhering to Python 3.7 standards. Specifically, we will grade you only using python 3.7.4. Besides the default libraries supplied in the standard Python distribution, you may use ONLY numpy and matplotlib libraries for this MP assignment. The time limit is set to 3 seconds for both problems.

Problem 1 [50 points]. Multilateration in 3D. You are to implement the function named `multilaterate(distances)` in the provided template python code in the file `multilaterate.py`. The function `multilaterate(distances)` takes a list of lists as its input. The list should have four landmark locations with associated (x, y, z, d) in formation in which x, y, z are the location of the landmark in 3D and d is the distance of the landmark to the unknown point to be localized. You should test your program to ensure it works with different input points correctly. Your program will be tested on several different setups and should provide reasonable output (i.e. for each coordinate of (x, y, z), either the absolute or relative error is less than 10^{-6}). `multilaterate.py` takes a single argument: the datafile name. Your program should then print out the location that is computed. A skeleton `multilaterate.py` is provided.

For your submission, besides the implementation, you need to provide a PDF file placed in the same folder to explain your implementation, i.e., how do you compute the location. This does not need to too mathematical, i.e., you can use mostly verbal descriptions to explain the main ideas behind your computation.

Problem 2 [50 points]. Implementation of a 2D Kalman filter. Suppose that there is a point mass moving in 2D with the system equation being

$$x_k = \begin{pmatrix} x_{1,k} \\ x_{2,k} \end{pmatrix} = \begin{pmatrix} x_{1,k-1} \\ x_{2,k-1} \end{pmatrix} + \begin{pmatrix} u_{1,k-1} \\ u_{2,k-1} \end{pmatrix} + \omega_{k-1},$$

and the observer being

$$z_k = \begin{pmatrix} z_{1,k} \\ z_{2,k} \end{pmatrix} = \begin{pmatrix} x_{1,k} \\ x_{2,k} \end{pmatrix} + \nu_k.$$

Assume that the system noise ω is a zero mean Gaussian with the covariance matrix

$$Q = \begin{pmatrix} 10^{-4} & 2 \times 10^{-5} \\ 2 \times 10^{-5} & 10^{-4} \end{pmatrix}$$

and the measurement noise is a zero mean Gaussian with the covariance matrix

$$R = \begin{pmatrix} 10^{-2} & 5 \times 10^{-3} \\ 5 \times 10^{-3} & 2 \times 10^{-2} \end{pmatrix}.$$

For your solution, you should provide

1. The Kalman filter update equations for this system (in a .pdf file). Put this file under the same folder as the other files for this problem.
2. A python file named `kalman2d.py` that you need to implement to process the given data (a skeleton file is provided). The python program takes four additional arguments. The first argument is the data file name. The second and third arguments are $\hat{x}_{1,0}$ and $\hat{x}_{2,0}$, respectively. The last argument is a value λ that will be used to compute P_0 as $P_0 = \lambda I$ in which I is the identity matrix. After getting the predicted $(x_{1,k}, x_{2,k})$ values, plot them using matplotlib as a set of 2D points and connect them sequentially using line segments. Do the same in the same plot to the observations (i.e., $(z_{1,k}, z_{2,k})$) using a different color.

A data file including the data below is also included in the problem folder. Your program should work for data that is different from this provided data set.

Table 1: Sample control input and observations for running the Kalman filter.

k	$u_{1,k-1}$	$u_{2,k-1}$	$z_{1,k}$	$z_{2,k}$
1	0.258286754	-0.323660615	2.275352754	0.973676385
2	0.167658082	-0.173943802	2.230101836	0.701036583
3	0.213711214	-0.096513636	2.77765305	0.664235947
4	0.257826742	-0.153303882	2.929903792	0.725217065
5	0.227367085	-0.226875214	3.192038877	0.604513851
6	0.29707499	-0.125012662	3.347010866	0.278565189
7	0.253380049	0.040957103	3.900306915	0.333608292
8	0.434690544	-0.179539834	3.95920546	0.266202459
9	0.233358557	-0.110426773	4.513029017	0.264158685
10	0.309397654	-0.039241497	4.681395671	0.128679188
11	0.268100576	-0.052630663	4.944550247	-0.183817474
12	0.336295529	0.050916321	5.380023776	0.314923847
13	0.383568488	-0.263521311	5.816818264	-0.115412464
14	0.271740175	-0.316833884	5.892258439	-0.677581348
15	0.37813998	-0.100109434	6.35130042	-0.551171782
16	0.32708519	-0.124288625	6.50389161	-0.955727407
17	0.318597789	-0.216869042	6.992389399	-0.951967449
18	0.215326213	-0.334744463	7.043205612	-1.221487912
19	-0.035040957	-0.077543457	7.006712655	-1.476399369
20	0.170197008	-0.062307673	7.076019663	-1.702334042
21	0.325171499	-0.008000277	7.534015162	-1.481437319
22	0.414100752	-0.043273351	8.001067914	-1.49148567
23	0.197609976	-0.391999135	8.211005889	-2.015420805
24	0.409390148	-0.114261647	8.649643037	-2.122464452
25	0.334974442	-0.065947082	8.925195479	-2.247646534

