Localization lab – Part 1
Work to be done

Questions with the symbol ® to the left are the questions you have to cover in your report.
Programs use global variables. **Clear the workspace before running any program** and close the graphs of the previous execution.
For the beginning of the lab: `clear all; close all; PlotRawData`
Later: `clear all; close all; MagnetLoc; PlotResults`
To compare graphs between executions, remove "`close all`".

Preparation

1. Retrieve the file `StudentDocs_Lab_1.zip` from your Moodle platform .
2. Create a folder for the lab and decompress the contents of the zip file in the folder.
3. Along the way, you may have to look back at the presentation of the lab (corresponding slides are in the zip file). Read the documents in the following order: the present document, than the "programs and data" document.
4. Using "`PlotRawData.m`", check the various paths corresponding to each data set. Don't forget to look at the speed data (figure 2). Figure 3 will be useful later.
5. The files "`EvolutionModel.m`" is missing (you have to write your own). It is called in `PlotRawData`, line 50. To write your own version, you have to clearly identify the definitions of X and U in "`PlotRawData.m`", which are the same as in "`MagnetLoc.m`". Make sure you get the exact same odometry results with your function and the provided one. To make sure it's your version that is run, rename "`EvolutionModel.p`" into something else, say "`EvolutionModel.backup`".
   Note: this task is elementary, but I want to make sure everyone is aware of what the evolution model exactly is.
6. "`MagnetLoc.m`" contains missing code which you need to provide. The missing code is replaced by "`***`". You can use the localization book to help you in this task. The solution is in the provided files, but use it only to check for calculation mistakes in your own solution. If you spot an error and do not understand why what you did is incorrect, ask the teacher.
7. ® Using figure 3 of "`PlotRawData.m`" and what you know about the sensor construction, evaluate the measurement noise variance. **You will submit a short report for this on Hippocampus/Aulaweb**, giving details of how you evaluate the variance. It can be handwritten, as long as it is clear. What is important here is to clearly justify the variance estimate. If you work in pairs, submit on the account of one of you only.
   To perform this task, I suggest you set the resolution of the encoders and sampling frequency to their maximum value (In `RobotAnsSensorDefinition.m`, set `dumbFactor` and `subSamplingFactor` to 1). Relevant information has been given in the presentation of the lab. A few tips:
   a) The noise is the difference between the idealized sensor model (the sensor is a line, the magnet a point) and reality.
   b) The two noises (x and y measurements) have very different origins and should not be mixed up. The noise in x is related to the diameter of the area in which the magnetic field is able to close a reed sensor. The noise in y is related to the spacing between reed sensors. To determine the variance of the noise in the x measurement, concentrate on figure 3 printed by `PlotRawData.m` (preferably for "diagonal45degrees", where all points of the robot travel the same distance and all reed sensors are used). For the noise variance in the y measurement, concentrate on the sensor construction.

c) You have to determine the error distribution to calculate the variances.

8. ® There is another important parameter to be set: the threshold for the Mahalanobis distance (`mahaThreshold`). It is set by using the Matlab `chi2inv` function or chi_square tables. You must understand what this function does and use it to determine a proper value for the threshold. The explanations given in the presentation of the lab should make this easy.
9. ® Once you have a measurement noise you consider reasonable, set the initial covariance matrix `Pinit`. `Pinit` is related to the uncertainties in setting the robot at the desired initial posture. The process has been described in the presentation of the lab. After that, the standard deviation `sigmaTuning` becomes your single tuning parameter. If the measurement noise value, Mahalanobis distance threshold and initial covariance matrix have been properly set, you should be able to find a proper value for `sigmaTuning`. **Submit** a report that explains the methodology to determine a proper value of the tuning parameter. Again, submit only once if you are working in pairs.
   For this tuning task, the parameters `dumbFactor` and `subSamplingFactor` in `RobotAnsSensorDefinition.m` should be reset to their original values, resp. 8 and 4.
   "Loop" trajectories are fundamental here, because the end position of the robot is known. Use "TwoLoops" for your tuning: it is the most demanding. Then make sure your tuning works with all datasets.
10. ® Starting with a correctly tuned filter, test the effect of over-estimating (resp. under-estimating) each of the following parameters and make sure you understand and can explain what happens. In particular, analyze how the term $CPC^t+Q\gamma$ evolves, and how it helps understand the behavior of the Kalman filter. To over-estimate a parameter, multiply its standard deviation by 10. To under-estimate, you can set its standard deviation to zero.
    a) Initial robot position variance.
    b) Measurement noise variance.
    c) The same test with sigmaTuning should have been done and understood during the tuning part of the lab, in question 9.
    d) For the initial position variance, also test the effect of a slight over-estimation. What is the limit, beyond which there are problems, and what are those problems?
11. ® Make sure you understand and can explain the results you obtain (not only the estimated path!) on all datasets. Each test has something to offer.
    a) The loop trajectories are instrumental to check that the algorithm works.
    b) Depending on variance determination and tuning, the twoLoops trajectory may have a fairly sharp correction around (x,y) point (60,260). Do you understand why?.
    c) The diagonal45degrees show two diverging straight lines. Do you understand what happened in this test?
    d) Check what happens with the line2magnets test when you correctly initialize the position to (0,27,0) and when you don't (it often happens that students start at 0,0,0, check what happens and make sure you understand).
    e) With line1magnet, on figure 3 compare the estimated standard deviation for the x and y components of the state. Why are they different? Compare figures 3 and 4. Why are they the same?
    f) With diagonal45degrees, on figure 3 compare the estimated standard deviation for the x and y components of the state. Why are they the same? Compare figures 3 and 4. Why are they different? Generally speaking, do the values in figure 4 vary significantly from one test to another? Do you understand why I plotted the standard deviations expressed in the robot frame? If it is not yet clear, compare figures 3 and 4 for the circles dataset.
    g) Check the estimated paths with the circles data set. Is something wrong with the EKF? Is it drifting out of control?