EL2320: Applied estimation

Diego González Morín {diegogm@kth.se}

Lab 1: EKF

1. Part I: Preparatory questions

1.1. Linear Kalman Filter:

Question 1: **What is the difference between a 'control' u_t, a 'measurement' z_t and the state x_t? Give examples of each?**

A control (u_t) modifies the state of the system; it triggers the transition between states. It can be defined as a signal that models the dynamics of our system, and it is applied to the previous state of the system to predict the current state. Measurements gather information from the real state of the system. They are used to update the current predicted state so it becomes more similar to the real one. The state represents the system in a certain time instant.

We could use a robot that moves forward and can measure the distance to the wall that is in front of it. The control would be the signal that commands the robot to move forward a certain distance. The state would be the position of the robot relative to the wall. The measurement would be the measured distance to this wall.

Question 2: **Can the uncertainty in the belief increase during an update? Why (or not)?**

No, we can explain this by using the Kalman Filter algorithm:

Prediction:

$$\overline{x_k} = Ax_{k-1} + Bu_{k-1}$$
$$\bar{P}_k = AP_{k-1}A^T + R$$

Update:

$$K_k = P_k H^T (HP_k H^T + Q)^{-1}$$
$$x_k = \bar{x}_k + K_k(z_k - H\bar{x}_k)$$
$$P_k = (I - K_k H)\bar{P}_k$$

The covariance matrix P_t represents the uncertainty in the belief. By definition, this covariance matrix has to be semidefinite positive, then we have that $(I - K_t H) < I$. As a consequence, we have:

$$Pt < \overline{Pt}$$

This mathematical conclusion has a logic explanation: even though our uncertainty increases during the prediction step as we are applying a control signal, this uncertainty decreases during the update as we gather information about the actual state of the system.

Question 3: **During update what is it that decides the weighing between measurements and belief?**

Kalman gain decides the weighing between the measurements and belief. Using the same equations as in the previous question we get that:

$$x_k = \bar{x}_k + K_k(z_k - H\bar{x}_k)$$

Our belief is represented by $H\bar{x}_k$ and the measurements by z_t. Then, the weight of the difference between this to parameters is given by the Kalman Gain K_t.

Question 4: **What would be the result of using a too large a covariance (Q matrix) for the measurement model?**

Let find first a mathematical answer for this question. We know that:

$$K_k = P_k H^T (HP_k H^T + Q)^{-1}$$

Then, as Q increases, the Kalman gain gets smaller. As we explained before, this parameter decides the weighing between the measurements and the belief. So the smaller this gain is, the less importance we give to the difference between the measurements and our prediction.

As a consequence, the measurements will almost not taken into account during the update state.

This is a logic conclusion if we analyse what the meaning of the Q matrix is; this matrix represents the covariance for the measurement model. The bigger the value of the covariance, the less we trust in our measurements. And as a consequence, we should take less into account these measurements during the update as they are not reliable enough.

The problem of using a too large Q matrix would be that the algorithm will take too much time to converge, as it is almost not using the measurements in the update step and the updated state will look similar to the predicted state.

Question 5: **What would give the measurements an increased effect on the updated state estimate?**

As it was explained before, the larger the value of the Kalman gain is, the more importance measurements are during the update state. The value of this gain is directly related to the value of the covariance matrix Q. Then, small values of Q mean that we have a high trust in our measurements and then the Kalman gain is higher.

Also, smaller values of H would increase the value of this gain, giving more weight to the measurements.

Question 6: **What happens to the belief uncertainty during prediction? How can you show that?**

The belief uncertainty is expressed by:

$$\bar{P}_k = AP_{k-1}A^T + R$$

Where $AP_{k-1}A^T$ and R are semidefinite positives. Then the uncertainty increases during the prediction state. This makes sense as during the prediction step, we are applying the control signal which inevitably adds uncertainty to our prediction as we are adding the process noise ε_t.

Question 7: **How can we say that the Kalman filter is the optimal and minimum least square error estimator in the case of independent Gaussian noise and Gaussian priori distribution? (Just describe the reasoning not a formal proof.)**

Minimum least square estimators try to reduce the Mean Square Error. In the case of the Kalman filter this is done as we try to decrease the covariance of the process on every time step. Then the Kalman filter is a minimum least square error estimator. As Kalman filter needs Gaussian noise and Gaussian priori distribution, we can assume the Kalman filter is the optimal one.

Question 8: **In the case of Gaussian white noise and Gaussian priori distribution, is the Kalman Filter a MLE and/or MAP estimator?**

Kalman filter can be considered both a MLE and a MAP estimator in the case of white noise and Gaussian priori distribution. It will be considered a MAP estimator if the initial state has a Gaussian distribution. On the contrary, if we don't have information about the initial state, its distribution would be uniform and then the Kalman filter will be consider a MLE estimator.

1.2. Extended Kalman Filter:

Question 9: **How does the extended Kalman filter relate to the Kalman filter?**

The extended Kalman filter is a modification of the linear Kalman filter so we can use the Kalman filter to non−linear systems. To use the extended Kalman filter we need to linearize the system.

The main modifications are:

- A matrix is substituted by the Jacobian Gt of the transition linearized state function.
- The measurement H matrix is substituted by the jacobian Ht of the linearized measurement function.

Question 10: **Is the EKF guaranteed to converge to a consistent solution?**

No. If our linearized model does not represent well the real non−linear system, the EKF is not guaranteed to converge. This usually happens to systems that are importantly not linear.

Question 11: **If our filter seems to diverge often can we change any parameter to try and reduce this?**

We can increase the noise added to the process R, so we decrease the effect of the inaccurate linearization. Also, if it diverges due to poor data associations, one option would be to modify the matching threshold.

1.3. Localization:

Question 12: **If a robot is completely unsure of its location and measures the range r to a known landmark with Gaussian noise what does its posterior belief of its location p(x, y, θ|r) look like? So a formula is not needed but describe it at least.**

As the robot only knows the range to the landmark (with some noise) its posterior belief of its location will look like a ring with radius r, as the robot can be in every possible point at a distance r from the landmark. As the measurement has noise, the belief would be a thick ring. The thickness of the ring depends on the noise of the measurement.

Question 13: **If the above measurement also included a bearing how would the posterior look?**

Now we would have a Gaussian distribution around the point defined by theta and r.

Question 14: **If the robot moves with relatively good motion estimation (prediction error is small) but a large initial uncertainty in heading θ how will the posterior look after traveling a long distance without seeing any features?**

As we have a high initial uncertainty in the heading, our posterior will have a C shape after traveling a long distance. The size of the C will be proportional to the distance travelled. Also, this C's thickness depends on the uncertainty of the noise of the process and the measurement.

Question 15: **If the above robot then sees a point feature and measures range and bearing to it how might the EKF update go wrong?**

As out posterior belief is a big C due to the initial uncertainty in the heading, when we read the range and heading to a landmark the main problem is that, as out posterior belief includes a lot of possible headings, we could identify the seen feature wrongly as another one and then the update of the localization would not be correct.

2. Part II: Matlab exercises

2.1. Warm up problem with standard Kalman Filter

Question 1: **What are the dimensions of εk and δk ? What parameters do you need to de need in order to uniquely characterize a white Gaussian?**

The system is defined by the following equations:

$$
\begin{aligned}
x_k &= \begin{bmatrix} p_k \\ v_k \end{bmatrix} \\
u_k &= a_0 \\
A &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \\
B &= \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \\
x_{k+1} &= Ax_k + Bu_k + \varepsilon_k \\
C &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\
z_k &= Cx_k + \delta_k
\end{aligned}
$$

We know that εk represents the noise in the position p_k and velocity v_k of the vehicle. As εk is added to Ax_k, they both have the same dimension. As x_k is a 2x1 vector, then εk is a

2x1 vector too. Using the same reasoning, we know that δk is the noise in the measurement z_k, then, both the measurement and the noise's dimension must agree. The measurement has the same dimension as Cx_k, which is 1x1. Then δk is a 1x1 vector. This is because, as C = [1 0], we are only measuring the position of the car.

Any Gaussian distribution is defined by $N(\mu,\Sigma)$ where μ is the mean vector and Σ is the covariance matrix. We would consider this Gaussian distribution a white Gaussian when it has zero mean value and the Σ is diagonal.

Question 2: **Make a table showing the roles/usages of the variables(x, xhat, P, G, D, Q, R, wStdP, wStdV, vStd, u, PP). To do this one must go beyond simply reading the comments in the code to seeing how the variable is used. (hint some of these are our estimation model and some are for simulating the car motion).**

Variable	Roles/Usages
x	Real state of the system. 2X1 vector: defined by position and velocity
xhat	Estimated state of the system. 2x1 Vector: defined by the estimated position and velocity. (Estimated by the Kalman Filter)
P	Is the covariance matrix of the estimate. It is a 2x2 matrix that represents the uncertainty of the current estimate.
G	Is a 2 dimension identity matrix. Used to scale the uncertainty of the state. No scale in this model
D	Is a 1 dimension identity matrix. Used to scale the uncertainty of the measurement.
Q	Uncertainty of the noise of the measurement. 1X1 dimension matrix.
R	Uncertainty of the noise of the state. 2x2 dimension matrix.
wStdP	Standard deviation of the noise in the position.
wStdV	Standard deviation of the noise in the velocity.
vStd	Standard deviation of the noise in the measurement.
u	Control signal. As the acceleration is constant, the control signal is 0 as we the speed is not changed.
PP	kx4 matrix. It stores the value of the uncertainty (covariance matrix) P of every step k.

Question 3: **Please answer this question with one paragraph of text that summarizes broadly what you learn/deduce from changing the parameters in the code as described below. Choose two illustrative sets of plots to include as demonstration. What do you expect if you increase/decrease the covariance matrix of the modelled (not the actual simulated) process noise/measurement noise 100 times (one change in the default parameters each time) for the same underlying system? Characterize your expectations. Confirm your expectations using the code (save the corresponding figures so you can analyse them in your report). Do the same analysis for the case of increasing/decreasing both parameters by the same factor at the same time. (Hint: It is the mean and covariance behaviour over time that we are asking about.)**

Before analysing the plot output for the different values of Q and R, we can guess how these plots will vary. The parameter that weights the belief is the Kalman gain. This Gain is expressed as follows:

```
K = P * C' * inv(C * P * C' + D * Q * D');
```

We can see that the K is inversely proportional to the value of Q. So the biggest the Q, the smaller is K and the less the filter 'trusts' in the belief, as:

```
xhat = xhat + K * (y - C * xhat);
```

Then, for bigger values of K, the value of the measurement has less weight in the update of the estimation. On the other hand, is Q is smaller, this would be mean that we trust more in the belief, and the measurement will be more considered in the update of the estimation.
If we analyse how R changes the output values, we have to take into account that the R is proportional to the value of the gain, so the bigger this values is, the bigger the weight of the measurements. This makes sense, as a bigger value of the variance of the state (R) means that our trust in the real process is low and we should trust the measurements more. In the same way, if R is smaller, it means that our trust in the process is high and we don't need to trust in the measurements as much.

After, running the code for different values of R and Q we get the following outputs:

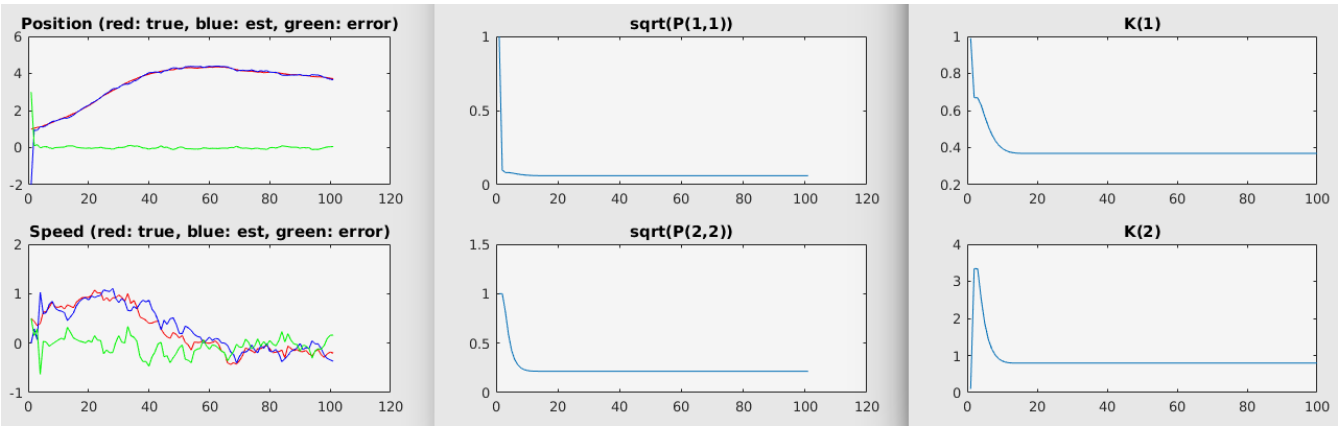

Figure 1: Normal. R' = R, Q' = Q.

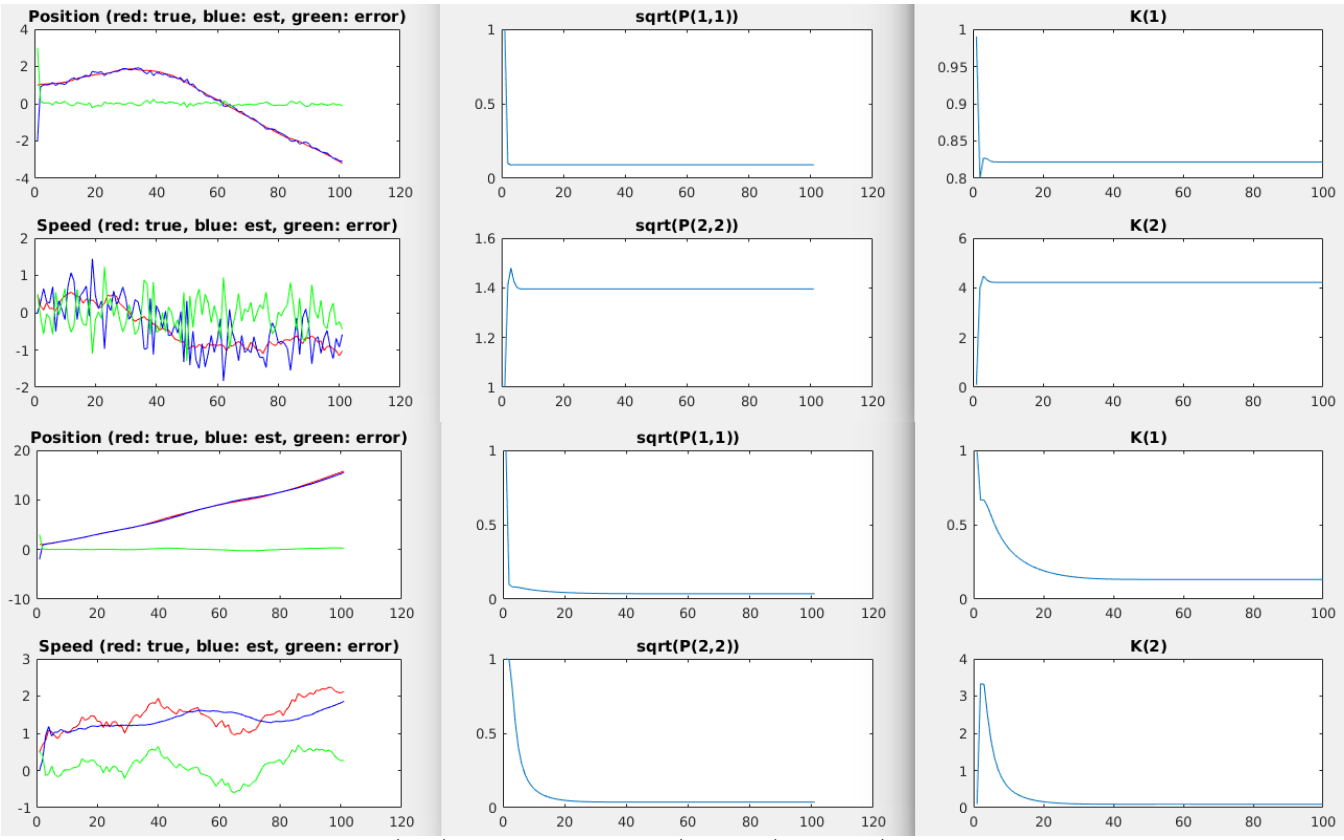

Figure 2: (top) R' = 100R, Q'=Q (bottom) R' = R/100, Q'=Q

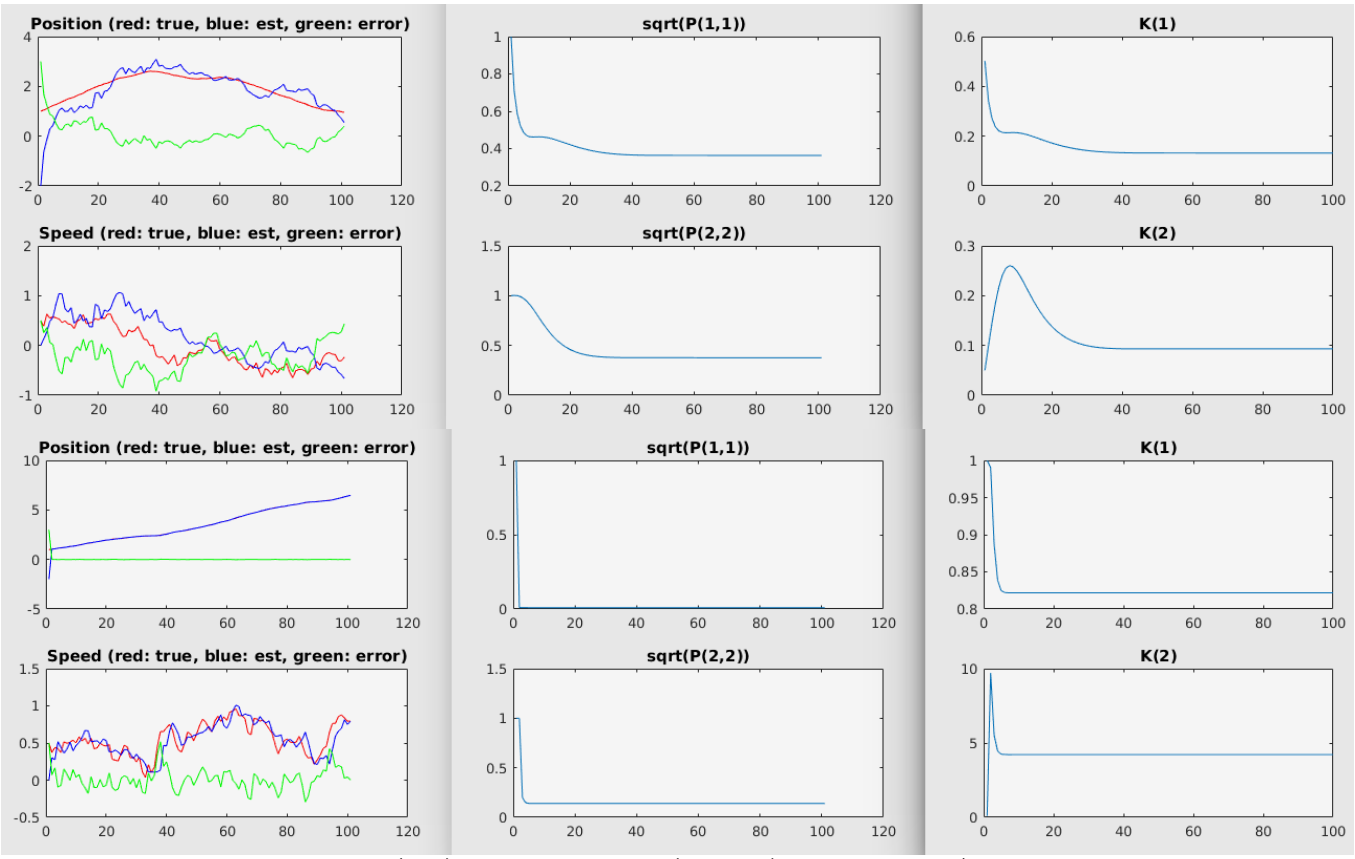

Figure 3: (top) R' =R, Q'=100Q (bottom) R' = R, Q'=Q/100

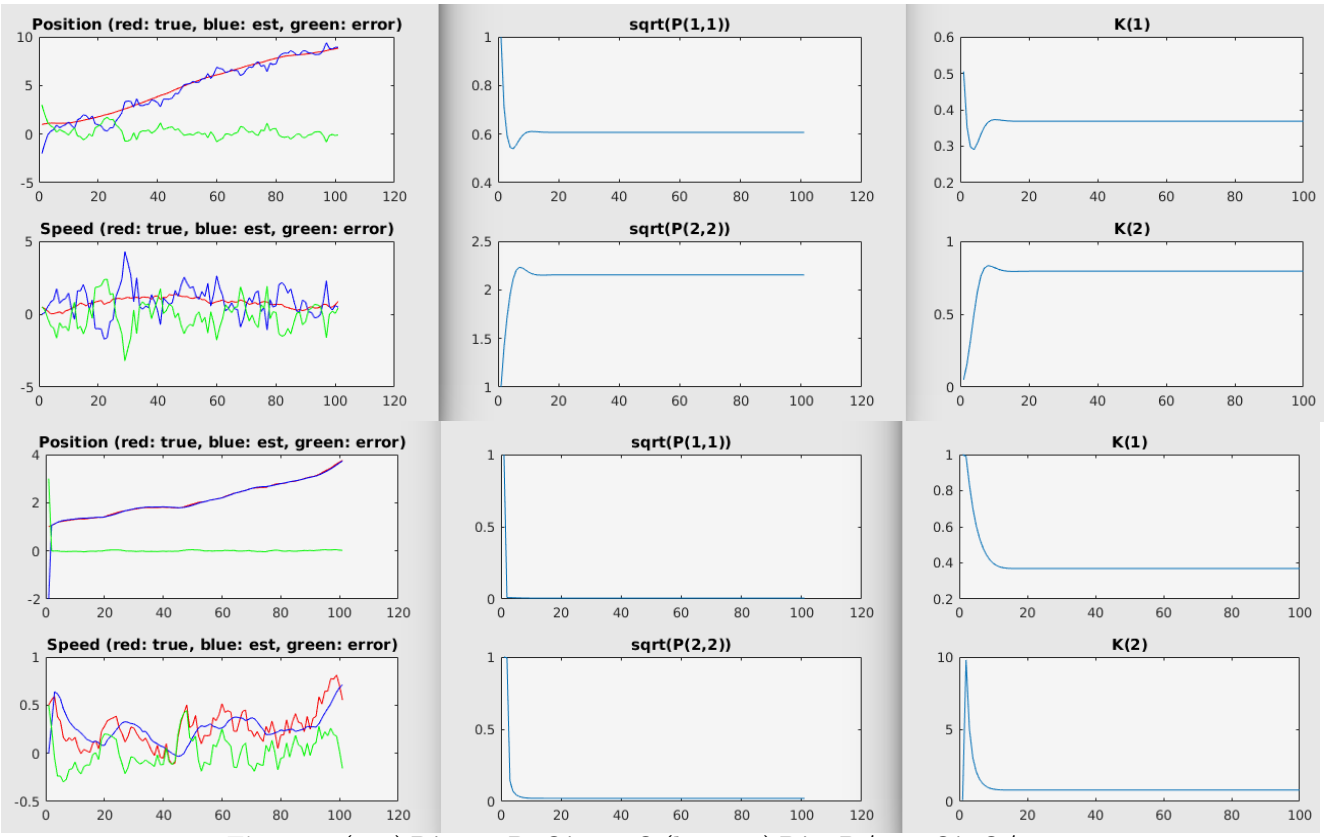

Figure 4: (top) R' =100R, Q'=100Q (bottom) R' = R/100, Q'=Q/100

As expected, we can see, comparing the figures 1 and 2, that the bigger the R, the bigger the K and the smaller the R the smaller the K. Also, for smaller of R, the variance of the estimate P gets closer to 0 than for bigger values of R. As for high values of R, our trust in the real process is lower and we give a lot of weight to the measurements, the oscillation of our estimation around the real values is greater.

Analysing figure 3 and 1 together, we also observe that are previous conclusions were correct. The bigger Q is, the less we trust in the measurement as we get smaller values of K. Also, obviously we get higher values of P, as the uncertainty is higher.

If we analyse the figure 4, we can observe, first, that the value of the K does not change a lot between higher values of Q and R and lower values. However, the dynamics of K are different.
The uncertainty of the model P increases for high values of Q and R, and decreases for lower values. This makes sense, as higher values of Q and R mean that we trust less both in the measurements and the estimation.

Question 4: How do the initial values for P and xhat affect the rate of convergence and the error of the estimates (try both much bigger and much smaller)?

In figure 5, and in comparison with figure 1, we can see that for higher initial value of P, it would mean a higher uncertainty in our estimation xhat, then we would need to give more weight to measurements, as our initial state is not trusted.
If P is lower, it would be mean that our trust in the initial state is high, and the weight given to the measurements would be smaller. The problem with a low initial value of P,

is that if our initial state is far from the real state, the filter will need a lot of time steps to converge.

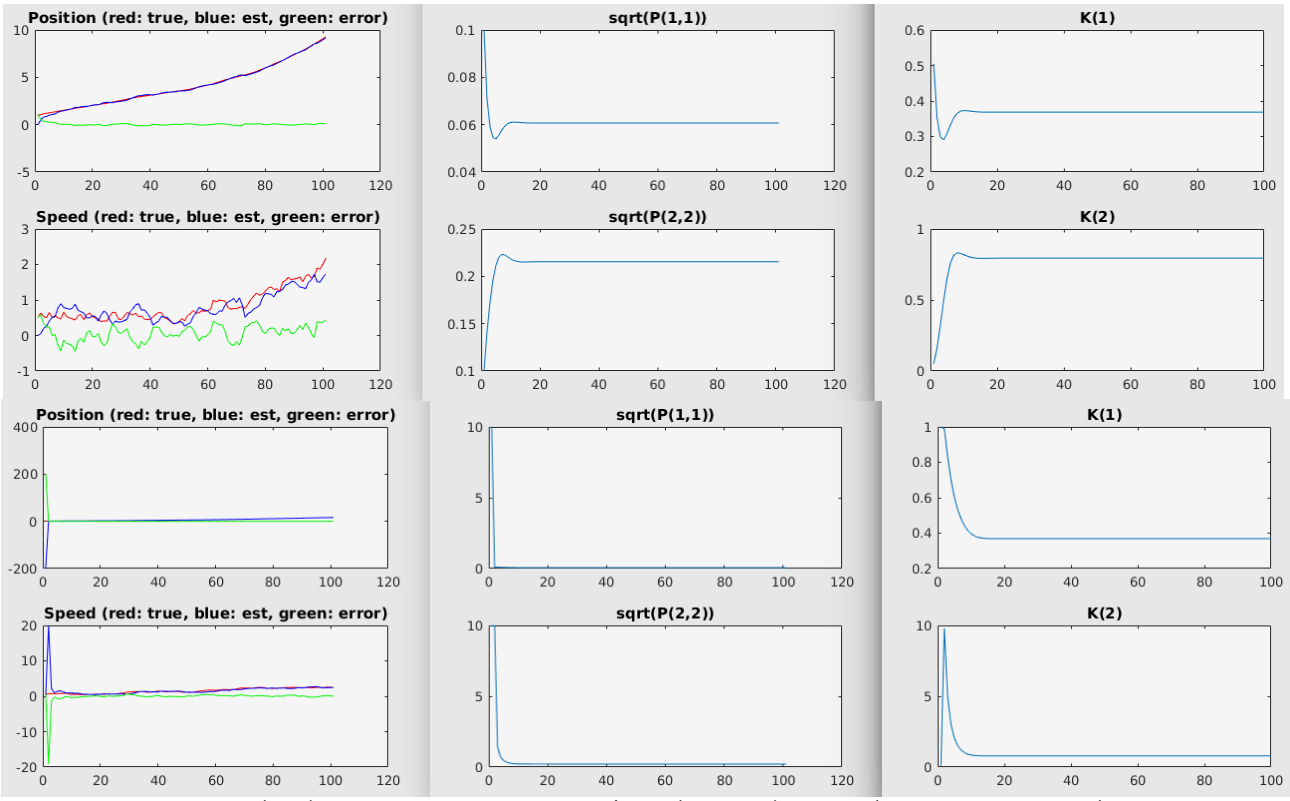

Figure 5: (top) P' = 100P, xhat' = xhat*100 (bottom) P' = P/100, xhat' = xhat/100

2.2. Main problem: EKF localization

Question 5: **Which parts of (2) and (3) are responsible for prediction and update steps?**

The equation (2) and (3) are expressed as:

$$\begin{cases} p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t}}, \bar{\mathbf{x}}_0, M) & = \eta p(\mathbf{z_t}|\mathbf{x_t}, M) \int p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) p(\mathbf{x_{t-1}}|\mathbf{z_{1:t-1}}, \mathbf{u_{1:t-1}}, \bar{\mathbf{x}}_0, M) \mathbf{dx_{t-1}} \\ p(\mathbf{x_0}|\bar{\mathbf{x}}_0) & = \delta(\mathbf{x_0} - \bar{\mathbf{x}}_0) \end{cases} \tag{2}$$

or equivalently

$$\begin{cases} bel(\mathbf{x_t}) & = p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t}}, \bar{\mathbf{x}}_0, M) = \eta p(\mathbf{z_t}|\mathbf{x_t}, M) \overline{bel}(\mathbf{x_t}) \\ \overline{bel}(\mathbf{x_t}) & = p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t-1}}, \bar{\mathbf{x}}_0, M) = \int p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) bel(\mathbf{x_{t-1}}) \mathbf{dx_{t-1}} \\ bel(\mathbf{x_0}) & = p(\mathbf{x_0}|\bar{\mathbf{x}}_0) = \delta(\mathbf{x_0} - \bar{\mathbf{x}}_0) \end{cases} \tag{3}$$

Then, observing the eq. (3) we can see that

$$\overline{bel}(\mathbf{x_t}) = p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t-1}}, \bar{\mathbf{x}}_0, M) = \int p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) bel(\mathbf{x_{t-1}}) \mathbf{dx_{t-1}}$$

This equation is the responsible for the prediction as it uses the Markov assumption to estimate the state x_t using only information from the previous state.
Then, the equation responsible for the update is:

$$bel(\mathbf{x_t}) = p(\mathbf{x_t}|\mathbf{u_{1:t}}, \mathbf{z_{1:t}}, \bar{\mathbf{x}}_0, M) = \eta p(\mathbf{z_t}|\mathbf{x_t}, M) \overline{bel}(\mathbf{x_t})$$

As it includes the measurement z_t.

Using the same reasoning, taking a look to the equation (2) we conclude that the part responsible for the prediction is:

$$\int p(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}}) p(\mathbf{x_{t-1}}|\mathbf{z_{1:t-1}}, \mathbf{u_{1:t-1}}, \bar{\mathbf{x}}_0, M) \mathbf{dx_{t-1}}$$

As it integrates all the previous states to predict the next state x_t.

Then, the update is added with the adition of:

$$= \eta p(\mathbf{z_t}|\mathbf{x_t}, M)$$

Question 6: **In the maximum likelihood data association, we assumed that the measurements are independent of each other. Is this a valid assumption? Explain why**

This assumption is correct if we have actual white noise; the sensor always behaves in the same way. This is: the sensor has zero mean noise that is uncorrelated between measurements, and then we can consider that all the measurements are independent of each other.

Question 7: **What are the bounds for δM in (8)? How does the choice of δM affect the outlier rejection process? What value do you suggest for δM when we have reliable measurements all arising from features in our map; that is all our measurements come from features on our map? What about a scenario with unreliable measurements with many arising from so called clutter or spurious measurements?**

As it is said in the lab notes, δ_M is a probability, then its bounds are [0, 1]. The higher the value of δM the higher the value we get for the threshold $Lamda_M$. It is explained in the lab notes that this is an upper threshold, so the higher this threshold is, less measurements will be rejected, only the ones in the extremes of the distribution. This can be translated as a high confidence in our measurements. In the same way, a low value of δM means that we reject more measurements as we are less 'confident' on them.

Question 8: **Can you think of some down−sides of the sequential update approach (Alg. 3)? Hint: How does the first [noisy] measurements affect the intermediate results?**

In the lab notes, point 3.2.7 we can find the principal down−side of the sequential update approach:

'The sequential update can cause problems if you get outliers into the system and you perform data association for every new measurement processed. After an update with an outlier the estimate might be thrown off and the uncertainty incorrectly reduced (this is called inconsistency, i.e. that the uncertainty does not capture the true situation) which means that the other measurements may be incorrectly discarded as outliers. The order in which the measurements are processed is therefore important in

the sequential update as in the sequential update; the association of the i^{th} measurement is computed using the estimates that are updated using the $i^{th} - 1$ measurement.'

The main problem then is that if we get noise in the measurement, it will affect the subsequent updates of the observations as they will be updated from a noisy estimate.

Question 9: How can you modify Alg 4 to avoid redundant re–computations?

The Alg. 4 is expressed in the figure 6. The main problem here is that we include inside the i loop (the one for the observations) the computation of $\hat{z}_{t,j}$, $H_{t,j}$, $S_{t,j}$. We have to calculate the value of this variables for every landmark. But we are doing this i times instead of just one. To avoid redundant re–computations then we should calculate $\hat{z}_{t,j}$, $H_{t,j}$, $S_{t,j}$ for every landmark before starting the loop for the observation. This way we only do this computations once instead of i times.

Algorithm 4 EKF Localization with Batch update for the i^{th} time step

for all Observations i in z_t **do**
  for all Landmarks j in M **do**

$$\begin{aligned}
\hat{\mathbf{z}}_{t,j} &= \mathbf{h}(\bar{\boldsymbol{\mu}}_t, M, j) \\
H_{t,j} &= H(\bar{\boldsymbol{\mu}}_t, M, j, \hat{\mathbf{z}}_{t,j}) \\
S_{t,j} &= H_{t,j}\bar{\Sigma}_t(H_{t,j})^T + Q \\
\boldsymbol{\nu}_t^{i,j} &= \mathbf{z_{t,i}} - \hat{\mathbf{z}}_{\mathbf{t,j}} \\
D_t^{i,j} &= (\boldsymbol{\nu}_t^{i,j})^T(S_{t,j})^{-1}\boldsymbol{\nu}_t^{i,j} \\
\psi_t^{i,j} &= \det(2\pi S_{t,j})^{-\frac{1}{2}} exp[-\tfrac{1}{2}D_t^{i,j}]
\end{aligned}$$

  end for

$$\begin{aligned}
\hat{c}_t^i &= \arg\max_j \ \psi_t^{i,j} \\
\hat{o}_t^i &= D^{i,c_t^i} \geq \lambda_M \\
\bar{\boldsymbol{\nu}}_t^i &= \boldsymbol{\nu}_t^{i,\hat{c}_t^i} \\
\bar{H}_{t,i} &= H_{t,\hat{c}_t^i}
\end{aligned}$$

end for

{For inlier indices $1, \ldots, n$}

$$\begin{aligned}
\bar{\boldsymbol{\nu}}_t &= \begin{bmatrix} (\bar{\boldsymbol{\nu}}_t^1)^T & (\bar{\boldsymbol{\nu}}_t^2)^T & \ldots & (\bar{\boldsymbol{\nu}}_t^n)^T \end{bmatrix}^T \\
\bar{H}_t &= \begin{bmatrix} (\bar{H}_{t,1})^T & (\bar{H}_{t,2})^T & \ldots & (\bar{H}_{t,n})^T \end{bmatrix}^T \\
\bar{Q}_t &= \begin{bmatrix} Q & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & Q \end{bmatrix}_{(2n\times 2n)} \\
K_t &= \bar{\Sigma}_t(\bar{H}_t)^T(\bar{H}_t\bar{\Sigma}_t(\bar{H}_t)^T + \bar{Q}_t)^{-1} \\
\boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + K_t\bar{\boldsymbol{\nu}}_t \\
\Sigma_t &= (I - K_t\bar{H}_t)\bar{\Sigma}_t
\end{aligned}$$

Figure 6: EKF localization with Batch update for the ith time step

Question 10: **What are the dimensions of vt and Ht in Alg 4? What were the corresponding dimensions in the sequential update algorithm? What does this tell you?**

Is easy to conclude, just observing the algorithm that the dimension for v_t is IxV where I is the number of obervations and V the size of the measurements vector (v_t is computed using the measurements). The size of H_t is IxVxM where M is the size of the state vector. The difference compared with the sequential update algorithm is that the computational cost is smaller as in the sequential algortihm the dimensions are V and VxM respectively.

2.3. Data sets

Scenario 1: **map_o3.txt + so_o3_ie.txt for Q = (0.01,1) m,degrees and R = 0.1 m,m,rad**

```
mean error(x, y, theta)=(0.000237, 0.000239, 0.000198)
mean absolute error=(0.003761, 0.004884, 0.002890)
total_time =8.514985
```

Figure 7: Output after running the code over the first scenario: map_o3.txt + so_o3_ie.txt for Q = (0.01,1) m,degrees and R = 0.1 m,m,rad. Green: true position, red: estimated, bkue: odometry

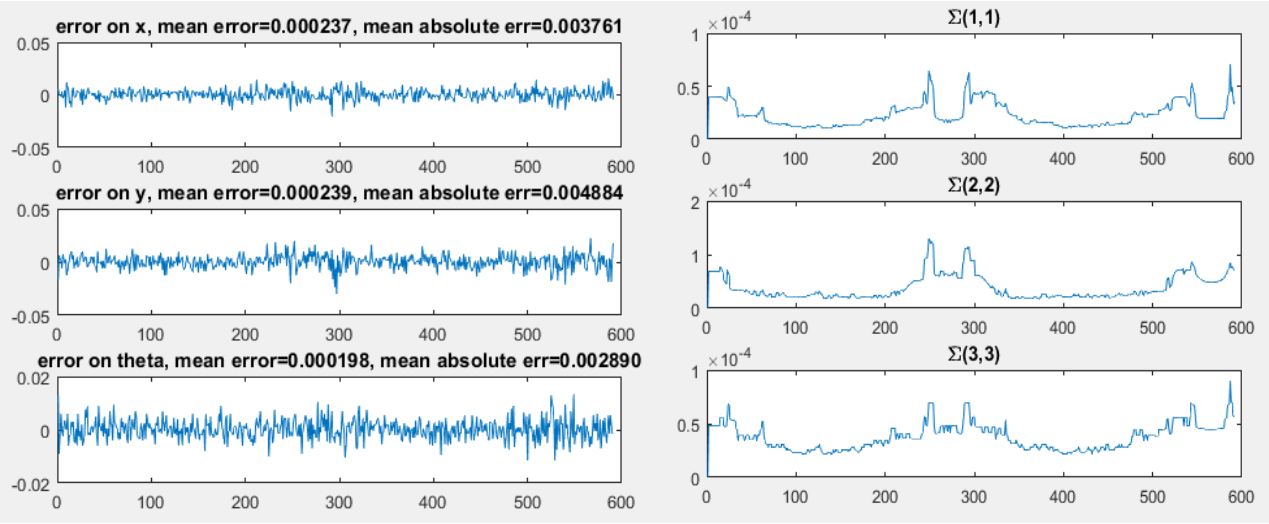

Figure 8: Output of the error and the variance after running the code over the first scenario: map_o3.txt + so_o3_ie.txt for Q = (0.01,1) m,degrees and R = 0.1 m,m,rad.

Scenario 2: map_pent_10_big.txt + so_pb_10_outlier.txt for Q = 0.2 m,degrees and R = 0.02 m,m,rad

```
mean error(x, y, theta)=(0.008749, 0.007580, -0.011826)
mean absolute error=(0.049479, 0.048008, 0.044477)
total_time =48.938197
```

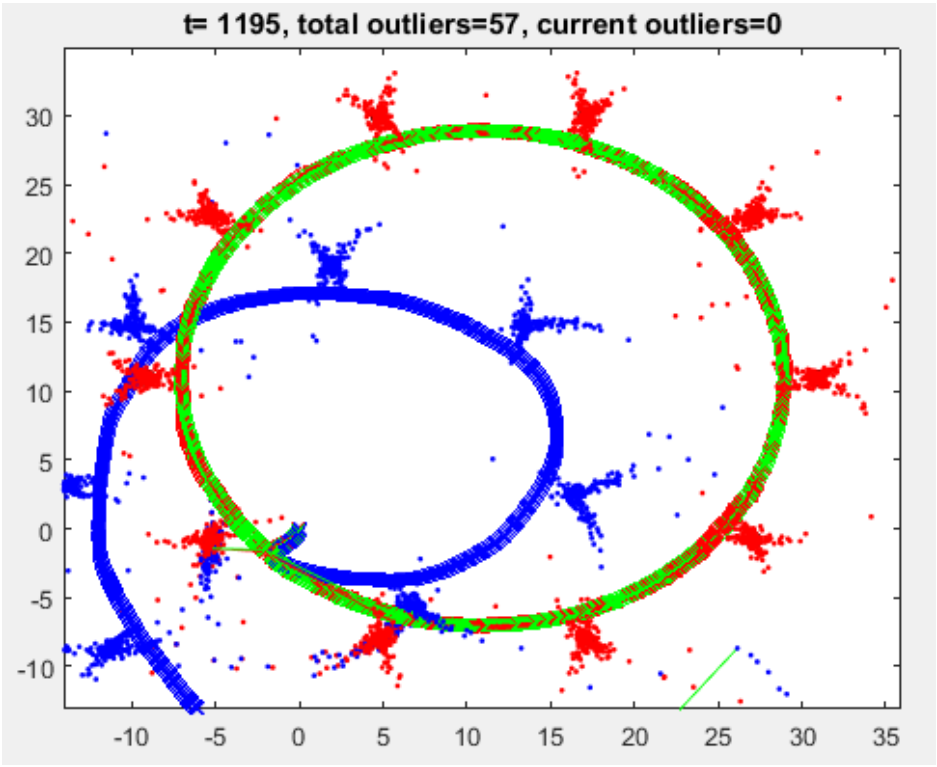

Figure 9: Output after running the code over the second scenario: map_pent_10_big.txt + so_pb_10_outlier.txt for Q = 0.2 m,degrees and R = 0.02 m,m,rad. Green: true position, red: estimated, bkue: odometry

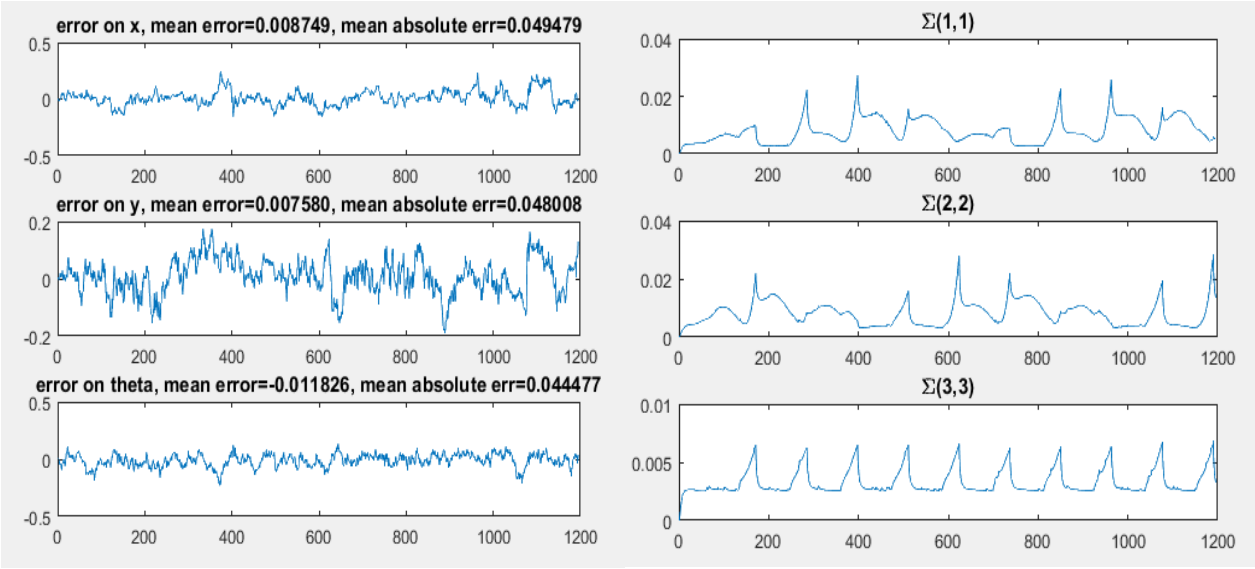

Figure 10: Output of the error and the variance after running the code over the second scenario: map_pent_10_big.txt + so_pb_10_outlier.txt for Q = 0.2 m,degrees and R = 0.02 m,m,rad.

Scenario 3: **map_pent_big_40.txt + so_pb_40_no.txt for Q = 0.1 m,degrees and R = 1 m,m,rad. Sequential update.**

```
mean error(x, y, theta)=(-2.618506, -1.733604, 0.439642)
mean absolute error=(4.462436, 4.795607, 0.440101)
total_time =3.671236
```

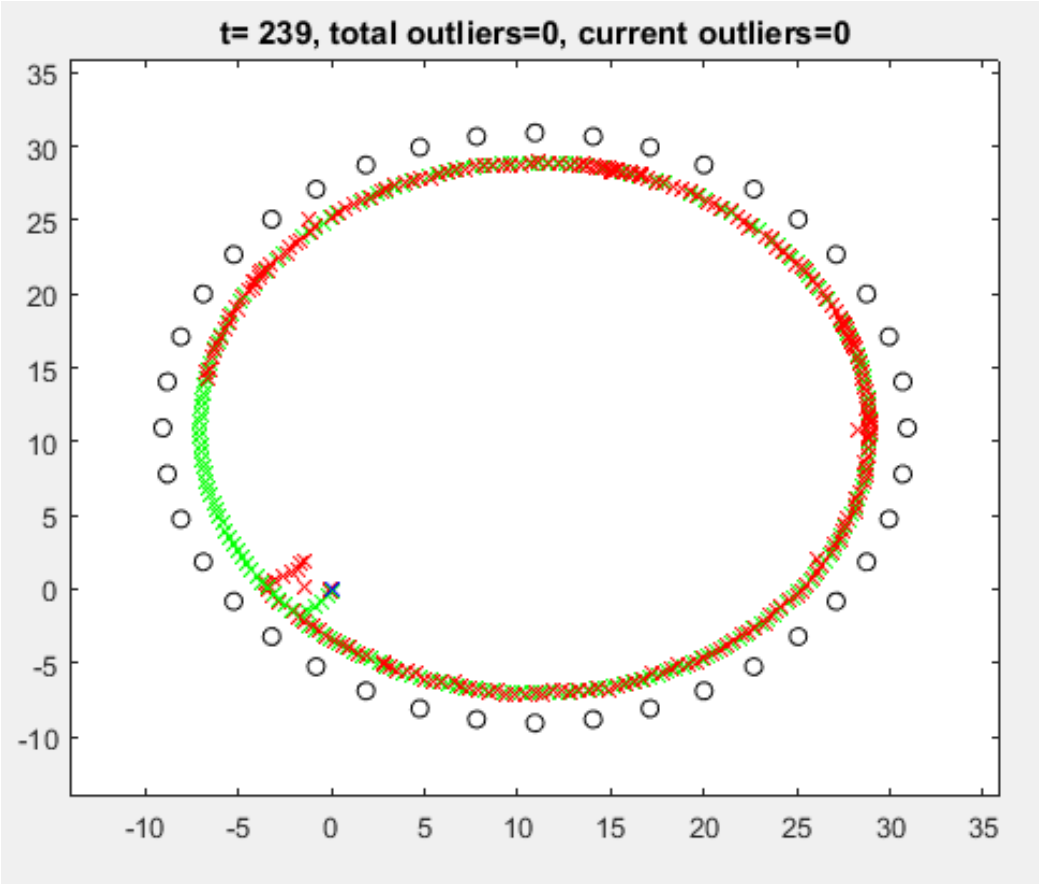

Figure 11: Output after running the code over the third scenario: map_pent_big_40.txt + so_pb_40_no.txt for Q = 1 m,degrees and R = 1 m,m,rad. Sequential update. Green: true position, red: estimated, bkue: odometry

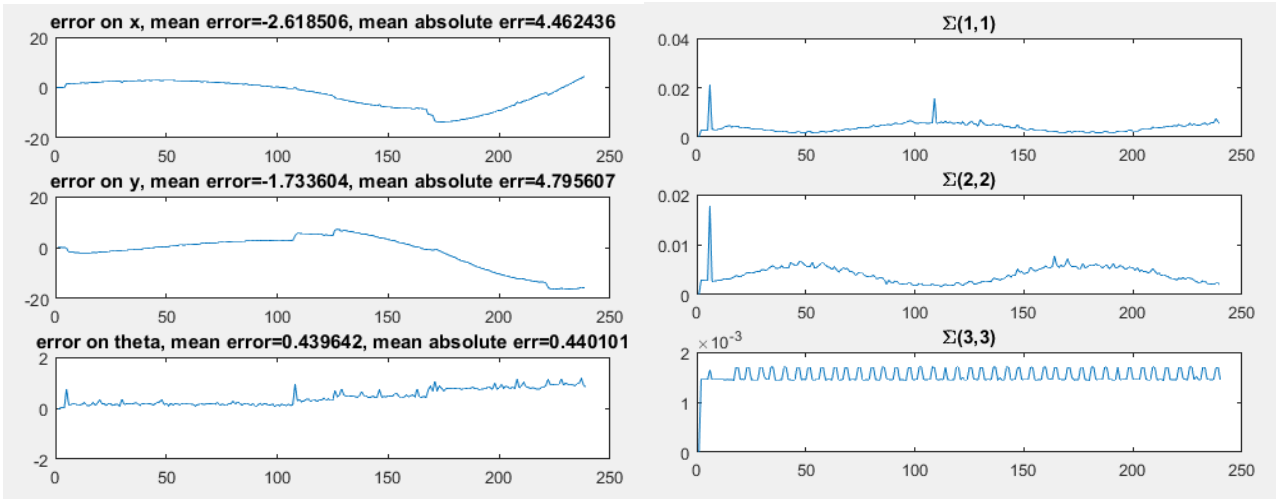

Figure 12: Output of the error and the variance after running the code over the third scenario: map_pent_big_40.txt + so_pb_40_no.txt for Q = 1 m,degrees and R = 1 m,m,rad. Sequential update.

Scenario 4: **map_pent_big_40.txt + so_pb_40_no.txt for Q = 0.1 m,degrees and R = 1 m,m,rad. Batch update.**

```
mean error(x, y, theta)=(-0.026019, -0.022660, 0.018390)
mean absolute error=(0.080351, 0.088318, 0.047883)
total_time =2.809077
```

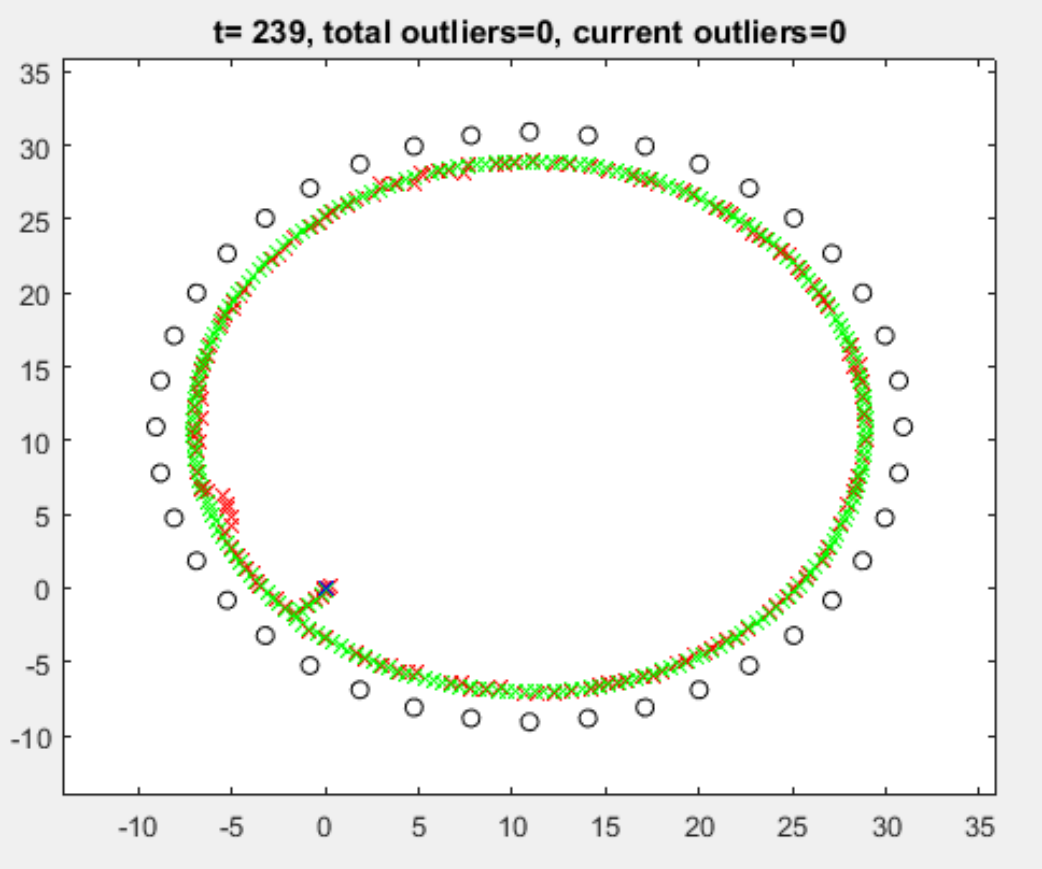

Figure 13: Output after running the code over the third scenario: map_pent_big_40.txt + so_pb_40_no.txt for Q = 1 m,degrees and R = 1 m,m,rad. Batch update. Green: true position, red: estimated, bkue: odometry

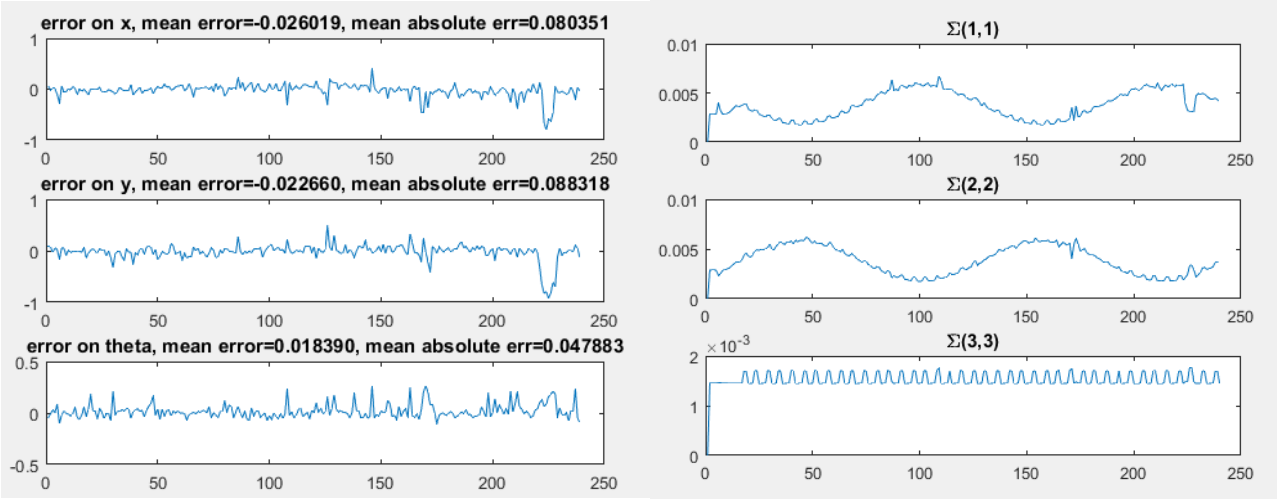

Figure 14: Output of the error and the variance after running the code over the third scenario: map_pent_big_40.txt + so_pb_40_no.txt for Q = 1 m,degrees and R = 1 m,m,rad. Batch update.