The Effects of Quantization and Intermittent Observations in Extended and Unscented Kalman Filter State Estimation

ELEC-E8105 Nonlinear Filtering and Parameter Estimation – Project work

Arponen Jani, 769684

I. Introduction

In this paper I present the project work for ELEC-E8105. My initial pitch for the project topic was to simulate filter performance in networked control systems, where data is transmitted over unreliable channels. In networked control systems, one or more of the control system signals is sent over a network, which introduce communication delay, quantization, packet losses and possibly scheduling conflicts – all of which affect state estimation and should be taken into account in filtering.

For my project, I chose to look into the effects of quantization and packet losses on state estimation through Extended and Unscented Kalman filters and subsequently the compounded effect of these estimators on control of the system. The reason for not using particle filtering for state estimation is rather simple: a big number of networked control systems are wireless, where the success of communication is related to the power of the transmitter. Many wireless control systems are also battery powered, and the communication (and it's overhead) is often the main power drain on the devices. Particle filters computational complexity can introduce so much power drain, that even with a better state estimate, the extra power could be better used on increased transmission power for less packet loss.

I have cited my sources as such: if the topic has been covered within the lectures, exercises, handouts or course book of ELEC-E8105, I have cited the course book [1]; if I have considered the information common knowledge to my peers in the course, then I have simply cited the Wikipedia article for the topic – as I may be unsure of where I have originally obtained this knowledge; if the information may be not common knowledge among my peers – I have tried to cite proper sources.

II. Materials

In this section, the groundwork of the project work is laid out from picking the system model, to developing a controller and the two state estimators for said model; and eventually doing some initial simulations. The outcome is a discrete time control system shown in block diagram form in Figure 1 – each section in the figure is explained in detail in the following subsections.

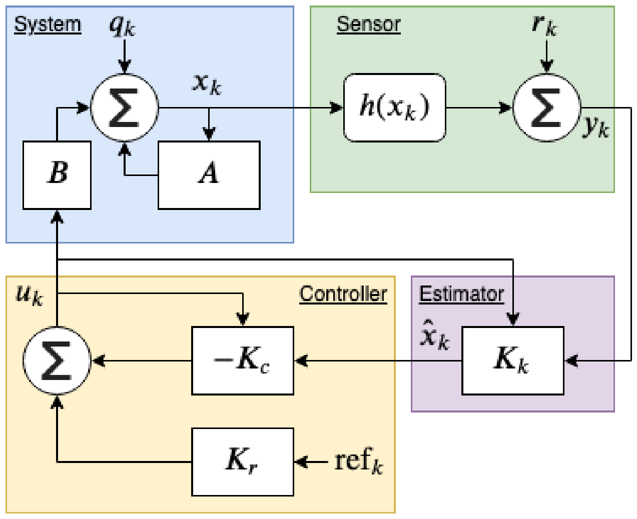

Fig. 1. Block diagram of the controlled system. K_k represents the state estimator.

A. System model

There were some (self imposed) criteria for picking a suitable system model for the project:

1) Non-linear system – due to the course topic.
2) Unstable system – a stable system will decay to the origin without input, thus lost measurements in networked control and is of less interest to me.
3) A simple system – due to time constraints, a simpler system will allow focus on the filtering and simulations over the complex model and controller structure.

There exist multiple models that fit the above criteria, but I ended up picking the bearings only tracking Wiener velocity model used in the course exercises 4-6 [1], as it is an LTI-model with a non-linear measurement equation and thus, a simple LQR could be used to control the system to given waypoints. The discrete system model is given by (1), where the system matrices are given in (2) and the used design parameters in Table I. The system and measurement noise coefficients were slightly reduced from the values given in the exercises in order to increase estimator and controller performance. Additionally, the tracked system is now controllable through B, where the speeds $\dot{x}, \dot{y}$ are directly modifiable by the control signal u_k. The idea is essentially that the vehicle that is being tracked gets its only measurements from the tracking sensors.

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + q_k \\ y_k &= h(x_k) + r_k \\ y_k^i &= \tan^{-1}\left(\frac{y - s_y^i}{x - s_x^i}\right) + r_k, \quad i = \{1, 2\} \\ q_k &\sim \mathrm{N}(0, Q), \quad r_k \sim \mathrm{N}(0, R) \end{aligned} \tag{1}$$

$$\begin{gathered} x_k = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}, \; A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\ Q = \sigma_q^2 \begin{bmatrix} \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^3}{3} & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & 0 & \Delta t & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & \Delta t \end{bmatrix}, \quad R = \sigma_r^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{gathered} \tag{2}$$

Parameter	Value	Description
Δt	0.1	Sampling time
σ_q^2	0.05	System noise, diffusion coefficient
σ_r^2	0.03	Measurement noise
s^1	$\{-1, 5\}$	Sensor 1 position
s^2	$\{5, 11\}$	Sensor 2 position

TABLE I
SYSTEM DESIGN PARAMETERS.

B. Controller

The system model (1) is LTI with a non-linear measurement equation. The output of the measurement equation can be estimated through the use of non-linear filters and thus, there is reason to use an infinite-horizon Linear Quadratic Regulator (LQR) as the controller for the system. LQR is a full state feedback controller, and it is the optimal controller with respect to the cost function (3), where C_Q, C_R and C_N are the cost weight matrices for the state, input and coupled costs respectively. [2]

$$J = \sum_{k=0}^{\infty} (x_k^T C_Q x_k + u_k^T C_R u_K + 2x_k^T C_N u_k) \tag{3}$$

The LQR control law is $u_k = -K_c x_k$, where the control gain matrix K_c that minimizes the above cost function (3) is given by (4) – where P is the solution to the discrete time algebraic Riccati equation (5).

$$K_c = (C_R + B^T PB)^{-1}(B^T PA + N^T) \tag{4}$$

$$\begin{aligned} P =& A^T PA + C_Q \\ &- (A^T PB + N)(R + B^T PB)^{-1}(B^T PA + N^T) \end{aligned} \tag{5}$$

For the system model (1), the LQR method with weights (6) produces the control gain matrix K_c (7). The input cost C_R was scaled up with respect to the state cost C_Q in order to slow down the controller response for smoother simulation.

$$C_Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; C_R = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}, \; N = 0 \tag{6}$$

$$K_c \approx \begin{bmatrix} 0.0917 & 0 & 0.1682 & 0 \\ 0 & 0.0917 & 0 & 0.1682 \end{bmatrix} \tag{7}$$

With K_c, the system is now able to control itself to the origin, but for the waypoint control a reference following controller is needed. This can be achieved with the control law $u_k = K_r \mathrm{ref}_k - K_c x_k$, where the reference gain K_r is calculated by (8) – where the C matrix is the linear models measurement matrix, i.e. only the positions x, y can be references as they are the ones measured.

$$\begin{aligned} K_r &= (C(I - A - BK_c)^{-1}B)^{-1} \\ C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ K_r &\approx \begin{bmatrix} -0.0917 & 0 \\ 0 & -0.0917 \end{bmatrix} \end{aligned} \tag{8}$$

Augmenting the system model (1) with the controller gives the augmented system model (9). Since only the states x, y are measured, a state estimator ($f_k(\dots)$ in the equations and the K_k-block in the Figure 1) is used for the control signal.

$$\begin{aligned} x_k &= Ax_{k-1} + u_{k-1} + q_k \\ y_k &= h(x_k) + r_k \\ \hat{x}_k &= f_k(\hat{x}_{k-1}, y_k, u_k, Q, R) \\ u_k &= -BK_c \hat{x}_k + K_r \mathrm{ref}_k \end{aligned} \tag{9}$$

C. Extended Kalman Filter

The Extended Kalman Filter algorithm for the augmented system model (9) can be written as below in its prediction (10) and update (11) steps – where the Jacobian H_k follows the partial derivatives given in (12). [1]

Predicted mean and covariance:

$$\begin{aligned} \hat{x}_k^- &= A\hat{x}_{k-1} + Bu_{k-1} \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned} \tag{10}$$

Updated mean and covariance:

$$\begin{aligned} e_k &= y_k - h(\hat{x}_k^-) \\ H_k &= \begin{bmatrix} \frac{\partial h(\hat{x}_k^-, s^1)}{\partial \hat{x}} & \frac{\partial h(\hat{x}_k^-, s^1)}{\partial \hat{y}} & 0 & 0 \\ \frac{\partial h(\hat{x}_k^-, s^2)}{\partial \hat{x}} & \frac{\partial h(\hat{x}_k^-, s^2)}{\partial \hat{y}} & 0 & 0 \end{bmatrix} \\ S_k &= H_x P_k^- H_x' + R \\ K_k &= P_k^- H_x^T S_k^{-1} \\ \hat{x}_k &= \hat{x}_k^- + K_k e_k \\ P_k &= P_k^- - K_k S_k K_k^T \\ \hat{x}_0 &\sim \mathrm{N}(x_0, P_0) \end{aligned} \tag{11}$$

$$
\begin{aligned}
h(x_k, s^i) &= tan^{-1}\left(\frac{y - s_y^i}{x - s_x^i}\right)\\
\frac{\partial h(x_k, s^i)}{\partial x} &= \frac{s_y^i - y}{(s_y^i)^2 - 2s_y^i y + (s_x^i)^2 - 2s_x^i x + x^2 + y^2}\\
\frac{\partial h(x_k, s^i)}{\partial y} &= \frac{x - s_x^i}{(s_y^i)^2 - 2s_y^i y + (s_x^i)^2 - 2s_x^i x + x^2 + y^2}
\end{aligned}
\tag{12}
$$

D. Unscented Kalman Filter

The Unscented Kalman Filter algorithm for the augmented system model (9) can be written as below in its prediction (14) and update (15) steps – where the matrix square root denotes Cholesky factorization, n is the number of states and the unscented transform weights and parameters are given in (13). Since the noises are Gaussian, I chose to leave the parameters as simple as possible with $\alpha = 1,\ \beta = \kappa = 0$. [1]

$$
\begin{aligned}
\lambda &= \alpha^2(n + \kappa) - n\\
W_0^{(m)} &= \frac{\lambda}{n + \lambda}\\
W_0^{(c)} &= \frac{\lambda}{n + \lambda} + (1 - \alpha^2 + \beta)\\
W_i^{(m)} &= \frac{1}{2(n + \lambda)}\\
W_i^{(c)} &= \frac{1}{2(n + \lambda)}\\
i &= 1, ..., 2n
\end{aligned}
\tag{13}
$$

$$
\begin{aligned}
&\text{Form sigma points:}\\
\chi_{k-1}^{(0)} &= \hat{x}_{k-1}\\
\chi_{k-1}^{(i)} &= \hat{x}_{k-1} + \sqrt{n + \lambda}\left[\sqrt{P_{k-1}}\right]_i\\
\chi_{k-1}^{(i+n)} &= \hat{x}_{k-1} - \sqrt{n + \lambda}\left[\sqrt{P_{k-1}}\right]_i\\
i &= 1, ..., n\\
&\text{Propagate through the dynamic model:}\\
\hat{\chi}_k^{(i)} &= A\chi_{k-1}^{(i)} + Bu_{k-1}\\
i &= 0, ..., 2n\\
&\text{Compute predicted mean and covariance:}\\
\hat{x}_k^- &= \sum_{i=0}^{2n} W_i^{(m)}\hat{\chi}_k^{(i)}\\
P_k^- &= \sum_{i=0}^{2n} W_i^{(c)}(\hat{\chi}_k^{(i)} - \hat{x}_k^-)(\hat{\chi}_k^{(i)} - \hat{x}_k^-)^T + Q
\end{aligned}
\tag{14}
$$

$$
\begin{aligned}
&\text{Form sigma points:}\\
\chi_{k-1}^{-(0)} &= \hat{x}_{k-1}^-\\
\chi_{k-1}^{-(i)} &= \hat{x}_{k-1}^- + \sqrt{n + \lambda}\left[\sqrt{P_{k-1}^-}\right]_i\\
\chi_{k-1}^{-(i+n)} &= \hat{x}_{k-1}^- - \sqrt{n + \lambda}\left[\sqrt{P_{k-1}^-}\right]_i\\
i &= 1, ..., n\\
&\text{Propagate through the measurement model:}\\
\hat{y}_k^{(i)} &= h(\chi_{k-1}^{-(i)})\\
i &= 0, ..., 2n\\
&\text{Compute updated mean and covariances:}\\
\mu_k &= \sum_{i=0}^{2n} W_i^{(m)}\hat{y}_k^{(i)}\\
S_k &= \sum_{i=0}^{2n} W_i^{(c)}(\hat{y}_k^{(i)} - \mu_k)(\hat{y}_k^{(i)} - \mu_k)^T + R\\
C_k &= \sum_{i=0}^{2n} W_i^{(c)}(\hat{\chi}_k^{(i)} - \hat{x}_k^-)(\hat{y}_k^{(i)} - \mu_k)^T\\
K_k &= C_k S_k^{-1}\\
\hat{x}_k &= \hat{x}_k^- + K_k(y_k - \mu_k)\\
P_k &= P_k^- - K_k S_k K_k^T
\end{aligned}
\tag{15}
$$

E. Simulation

With the system now explained through the augmented model and the two state estimators, we can take a look into simulating the system in MATLAB. As previously mentioned, the system controls to waypoints, e.g. with the waypoints in (16) the system will attempt to follow a square pattern.

$$
\text{ref}_i = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}10\\0\end{bmatrix}, \begin{bmatrix}10\\10\end{bmatrix}, \begin{bmatrix}0\\10\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right\} \tag{16}
$$

Determining whether a waypoint is reached was done with simple threshold values for norm to the center of the waypoint. Additionally, a similar threshold was added on the velocity estimates, such that the system has properly slowed down and ”visited” the waypoint, before moving on to the next one. Both threshold values were picked arbitrarily through trial and error and I settled on 0.2 units for the reference and 0.5 units for the velocity estimate. The positioning of the sensors was also picked arbitrarily and some benefits could be gained from better positioning.

To evaluate the state estimator performance, Root Mean Squared Error (RMSE) (17) between the true state and the estimate was used, mostly interested in the position x, y error. To evaluate the controller performance, total quadratic cost scaled by the LQR input cost matrix (18) was used. The last performance parameter is the count of total time steps until the last waypoint was reached, which indicates a combination of the state estimate and controller performances.

$$
e_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{1:N} - \hat{x}_{1:N})^2} \tag{17}
$$

$$J_u = \sum_{i=1} \sum_{j=1} \left[u_{1:N} u_{1:N}^T C_R\right]_{i,j} \tag{18}$$

The system was simulated with 3 different schemes – all with the same seed: 1. feeding the true state directly to the controller and the EKF and UKF simply estimate the trajectory, this can be seen in Figure 2. 2. feeding the EKF state estimate to the controller, this can be seen in Figure 3. 3. feeding the UKF state estimate to the controller, this can be seen in Figure 4. The performance parameters mentioned earlier were collected to Table II. From the figures and the performance parameters we can see, that overall the EKF is a better filter, be that for the purely tracking or controlling solution. Even though the seed is locked, the control signal will obviously affect the systems differently and thus the values here aren't a perfect representation of what could be the ground truth over multiple simulations.

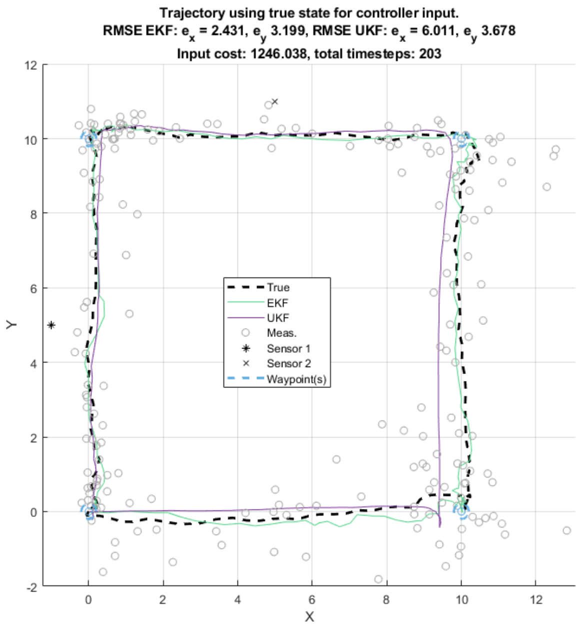

Fig. 2. System simulation, controller is fed the true state directly. EKF and UKF are just estimating the trajectory.

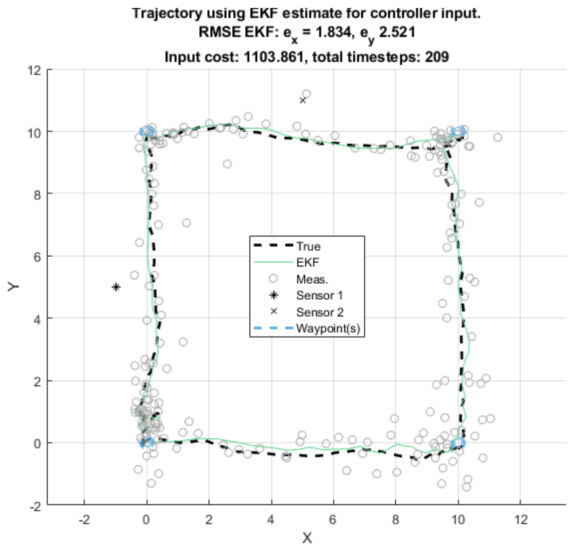

Fig. 3. System simulation, controller is fed EKF state estimate.

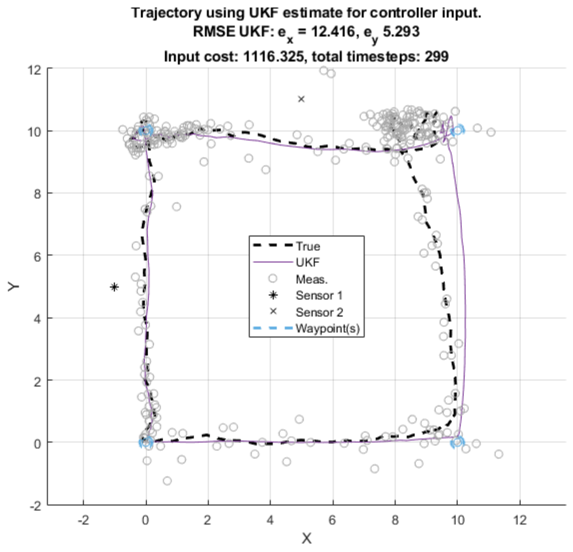

Fig. 4. System simulation, controller is fed the UKF state estimate.

Estimator	Parameter	Value	Description
Direct	$e_{x_{EKF}}$	1.858	Pos. x rmse
	$e_{y_{EKF}}$	2.276	Pos. y rmse
	$e_{x_{UKF}}$	3.095	Pos. x rmse
	$e_{y_{UKF}}$	2.458	Pos. y rmse
	J_c	1246.0	Input cost
	N	203	Timesteps
EKF	$e_{x_{EKF}}$	1.834	Pos. x rmse
	$e_{y_{EKF}}$	2.521	Pos. y rmse
	J_c	1103.9	Input cost
	N	209	Timesteps
UKF	$e_{x_{UKF}}$	12.416	Pos. x rmse
	$e_{y_{UKF}}$	5.296	Pos. y rmse
	J_c	1116.3	Input cost
	N	299	Timesteps

TABLE II
BASELINE VALUES FOR TRACKING AND EKF/UKF CONTROL.

To me, the interesting bit from the initial simulations is that the UKF controlled system was almost 50% slower to reach the final destination, but as a result spent the least amount of input resources per time step. This seemed to hold for other seeds as well. It could simply be a side effect of the poor position estimate leading to more time spent reaching waypoints and only a small amount of input resources used close to the waypoint.

III. METHODS

In this section, the control system developed in the last section is expanded on by turning it into a networked one with measurement quantization and packet losses induced by a noisy channel. The resulting networked control system can be seen in block diagram form below in the Figure (5). Each addition compared to the earlier described control system is explained in the following subsections and additional motivation is given for each.

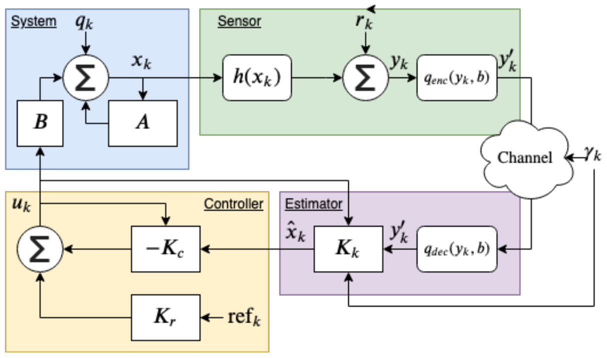

Fig. 5. Block diagram of the networked control system.γ_k represents the realization of successful measurement.

A. *Quantizing measurements*

Quantizing measured values is a reality in modern digital control systems. Typical analog-to-digital converters (ADC) on microcontrollers have in the order of 8 to 16 bits of resolution over their measurement range [3], which in itself doesn't necessarily introduce much measurement error, as the range can be vastly smaller than the floating point numbers used to represent the measurement, i.e. a sensor with a sufficiently small measurement range compared to the floating point data types used to represent the value in the microcontroller. However, this is not always what is meant by quantization in networked control systems. A network channel is able to support a finite amount of bandwidth, meaning it has a cap on the bitrate of data that can be transferred. In networked systems, many subsystems may need to transmit data over the same channel and thus, the amount of communication over said channel should be kept to a minimum, which isn't always feasible. Wireless channels can experience fading, where the probability of a successful transmission drops, sometimes dramatically, which can be combated by reducing the overall bitrate. With these reasons, and many more that were not explained here, quantizing already measured signals to a lower standard can be used to convey some information to the control system, when the other option is zero information due to unsuccessful transmission. [4], [5]

The quantizing strategy that I went with for this project is the simplest possible: uniform quantization where the encoder on the sensor end and the decoder on the estimator end agree on a bitlength N_{bits} and minimum and maximum values prior to the networked control system being switched on. With uniform quantization, the measurement range is split into $2^{N_{bits}}$ equal sized chunks that are rounded to [6]. When both ends know the quantization parameters, the actual limits do not need to be transmitted with each message (although this could be done in adaptive quantizing algorithms). The sensor positions given in table I result in measurements as shown in the below Figure 6 when sweeping over the x, y positions that describe the square of the waypoints in (16). The angle measurement for both sensors stays between $\frac{-4}{5}\pi ... \frac{1}{5}\pi$, so I used this range as the quantization minimum and maximum values.

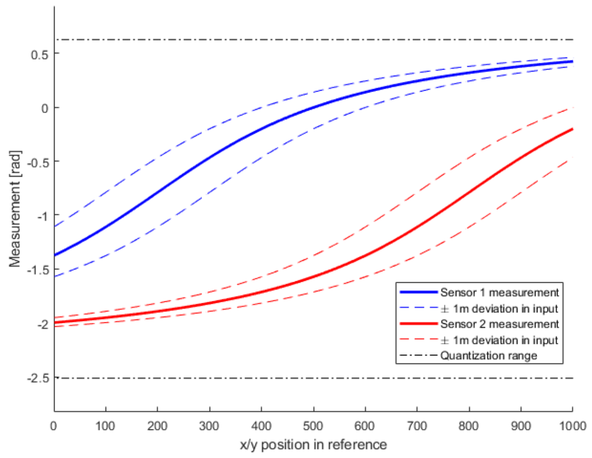

Fig. 6. Angle measurements for sensor positions and the fixed quantization range is visualized.

Quantization introduces error to the measurement. In the case of using a uniform quantization, this error can be thought of as a uniform distribution – the probability of the measured value being turned into the quantized one is uniform. From the point of view of the state estimation, this can be included in the innovation covariance calculation of the Kalman filters for the linear cases. However, this is not optimal as the Kalman filters are designed for Gaussian noises, but for sufficiently small quantization values i.e. $\sigma_{quant} << \sigma_{noise}$ this is negligible – which makes sense as it happens everywhere in digital control systems. The uniform distribution of the quantizer has error (19) and standard deviation σ_{quant} (20) [4], [6]. The effect of the quantization distribution can be seen in Figure (7).

$$-\frac{2^{-N_{bits}}}{2} \leq e_q \leq \frac{2^{-N_{bits}}}{2} \tag{19}$$

$$\sigma_{quant} = \frac{2^{-N_{bits}}}{\sqrt{12}} \tag{20}$$

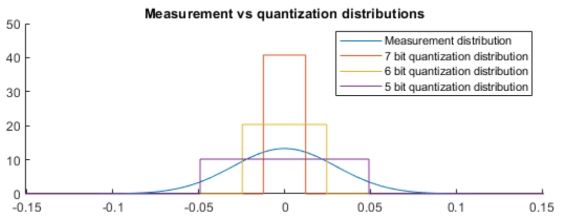

Fig. 7. Measurement distribution compared to low bitlength quantization distributions. The quantization here is done between $-\pi \ldots \pi$.

As mentioned earlier, the quantization error can be handled in the innovation covariance calculation of the linear Kalman filter algorithms by simply adding it to the measurement noise. Technically, the non-linear quantization is part of the measurement function and should be included there for the non-linear filters, but the Extended Kalman filter won't be able to deal with it as forming of the measurement Jacobian H_x requires differentiability, which is not possible with the introduced quantization "jumps". As such, I have approached the EKF case with the same idea as the linear filter case and the quantization is done in the innovation covariance calculation step (21).

$$S_k = H_x P_k^- H_x' + R + \begin{bmatrix} \sigma^2_{quant} & 0 \\ 0 & \sigma^2_{quant} \end{bmatrix} \tag{21}$$

The Unscented Kalman filter does not have the same differentiability requirements and as such, I added the quantization to the update step of the filter in sigma point propagation (22) and also to the corresponding innovation covariance calculation (23).

$$\begin{aligned} \hat{y}_k^{(i)} &= q_{enc}(h(\chi_{k-1}^{-(i)}), \min, \max, N_{bits}) \\ i &= 0, ..., 2n \end{aligned} \tag{22}$$

$$S_k = \sum_{i=0}^{2n} W_i^{(c)} (\hat{y}_k^{(i)} - \mu_k)(\hat{y}_k^{(i)} - \mu_k)^T + R + \begin{bmatrix} \sigma^2_{quant} & 0 \\ 0 & \sigma^2_{quant} \end{bmatrix} \tag{23}$$

B. Remote state estimation

Networked control systems are concerned with control systems where one or more of the control signals are sent over a network channel. The communications introduce delay and sometimes packet loss, when channels are not ideal. This is most evident in wireless channels, which can experience periodic or environment based fading, where the probability of a successful transmission can vary greatly. Network congestion and transmission scheduling conflicts may also introduce lost packets, but these may happen even without loss of channel quality. There are many different models for these channels, but for this project, I chose to focus on the simplest possible – a memoryless channel, which abstracts all of the different effects with a single constant packet loss probability p_{loss}. [7]

By incorporating results from information theory, researchers [8] have come up with the optimal result for state estimation in linear Gaussian models with intermittent measurements, by augmenting the Kalman filter equations (25) with $\gamma_k \in \{0, 1\}$. This new term signifies the success of transmitting the measurement over the channel and follows (24). In real systems, it is easy to reason how γ_k is available as the control system is expecting a periodic message, if one does not arrive, it is deemed lost.

The results of (25) can be thought of as a sort of feed forward control when a measurement is lost and the uncertainty in the state estimate increases for each lost measurement. It can be easily shown that for unstable systems the covariance P_k grows exponentially, but each successful measurement rapidly reigns in the state estimate. An example of this is shown in Figure 8, where the error bars visualize the growth of uncertainty in the state estimate.

$$\begin{cases} p(\gamma_k = 1) & = 1 - p_{loss} \\ p(\gamma_k = 0) & = p_{loss} \end{cases} \tag{24}$$

$$\begin{aligned} &\textbf{Predict:} \\ &\quad \hat{x}_k^- = A\hat{x}_{k-1} + B_k u_{k-1} \\ &\quad P_k^- = A P_{k-1} A^T + Q_k \\ &\textbf{Update:} \\ &\quad S_k = C_k P_k^- C_k^T + R_k \\ &\quad K_k = P_k^- C_k^T S_k^{-1} \\ &\quad \hat{x}_k = \hat{x}_{k-1} - \gamma_k K_k (y_k - C_k \hat{x}_k^-) \\ &\quad P_k = P_k^- - \gamma_k K_k S_k K_k^T \\ &\textbf{Control:} \\ &\quad u_{k+1} = -K_c \hat{x}_k + K_r \mathrm{ref}_k \end{aligned} \tag{25}$$

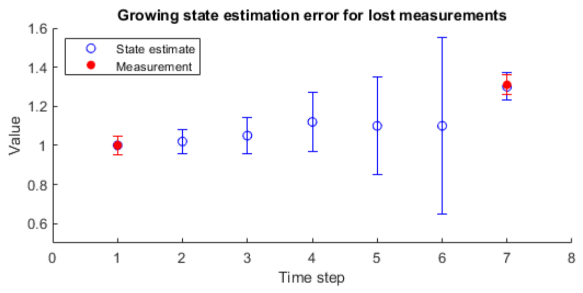

Fig. 8. Lost measurements effect on the uncertainty of state estimate visualized through error bars.

Extending the linear approach (25) to remote state estimation for the EKF (10,11) and UKF (14,15) by augmenting the update steps with γ_k gives us finalized filter algorithms for this project.

IV. Simulation and Results

In this section I show results for the simulated models under different quantization and packet loss schemes. In the simulations, I am modeling both of the sensors with a single channel, i.e. both measurements are lost in the event of a lost transmission, even though in a more realistic model, these would most likely be their own transmissions.

A. High quantization

Very high quantization could not be used the waypoints described in (16), so I chose to look into this through a simple structure of starting from point $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and controlling to point $\begin{bmatrix} 10 \\ 10 \end{bmatrix}$. The results can be seen from Figures 9 - 11, which show the system's response with a 6-bit quantizer on the measurements and no packet loss. Varying the seed gave different results, but in general, the UKF performed much better than EKF in both the state estimate RMSE and input cost metrics.

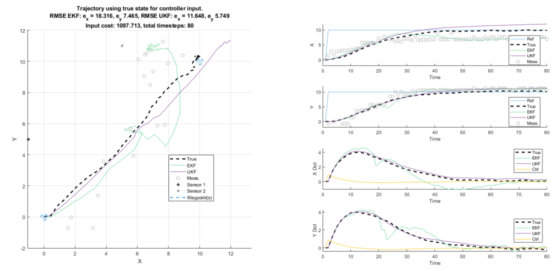

Fig. 9. 6 bit quantizers effect on trajectory estimation, with directly fed state control.

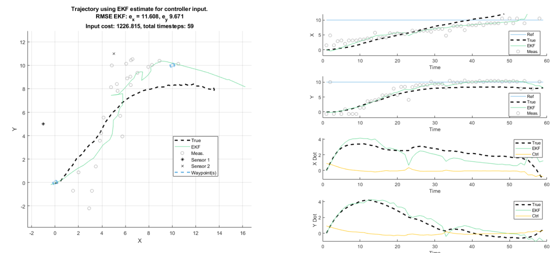

Fig. 10. 6 bit quantizers effect on control over an Extended Kalman filter state estimator.

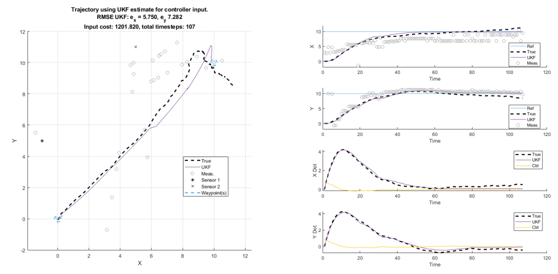

Fig. 11. 6 bit quantizers effect on control over an Unscented Kalman filter state estimator.

B. High packet loss

The packet loss simulations could be done against the same waypoints (16) as the baseline simulations earlier and thus can be at least somewhat compared. Once again, the seed used for the simulations affects the results here, even more so than for any of the other simulations before perhaps, as a string of lost measurement close to a waypoint contains more information than losing every other for the entire trajectory. Running the simulations for multiple seeds, it seemed that roughly 65% seemed to still lead to a successful traversal of the waypoints by the true state, which can be seen in Figures 12-14. Overall, the EKF seemed to perform better in terms of state estimate, even at up to 85% packet loss, but this wasn't entirely repeatable. Since the system model (1) is very simple, with a fully feed forward system ($p_{loss} = 100\%$), the obvious result is that the estimates follow the waypoints directly and the both of the filters produce the same (or very close to same) results and deviance from the true state, as the seed is locked.

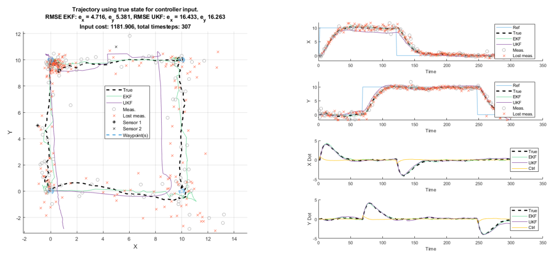

Fig. 12. System controlled by direct state with 65% packet loss.

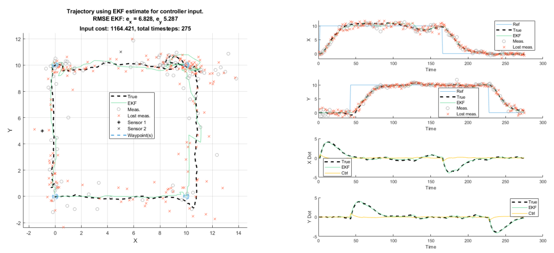

Fig. 13. System controlled by EKF state estimate with 65% packet loss.

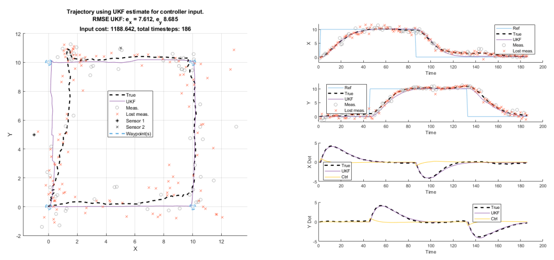

Fig. 14. System controlled by UKF state estimate with 65% packet loss.

C. Sweeping results

I was also interested in doing some data collection for different packet loss probabilities. I chose to modify the reference waypoints to a sort of semi circle (26), such that feed forward would play less of a role as the waypoints change more frequently. For these waypoints, I ran the simulation for increasing $p_{loss} \in \{0\%, 5\%, ..., 50\%\}$ for multiple seeds and averaged them out to get a sense of performance parameters as a function of p_{loss}. An example of a simulation for the semi circle track can be seen in Figure 15 with $p_{loss} = 10\%$. Now this is only a single sample from the different seeds, and the EKF is also having some trouble around the first waypoint.

$$\begin{aligned}\text{ref}_i = \{ &\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}10\\0\end{bmatrix}, \begin{bmatrix}9.5\\2.1794\end{bmatrix}, \begin{bmatrix}8.5\\3.5707\end{bmatrix}, \begin{bmatrix}7\\4.5826\end{bmatrix}, \\ &\begin{bmatrix}5\\5\end{bmatrix}, \begin{bmatrix}3\\4.5826\end{bmatrix}, \begin{bmatrix}1.5\\13.5707\end{bmatrix}, \begin{bmatrix}0.5\\2.1794\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix} \}\end{aligned} \qquad (26)$$

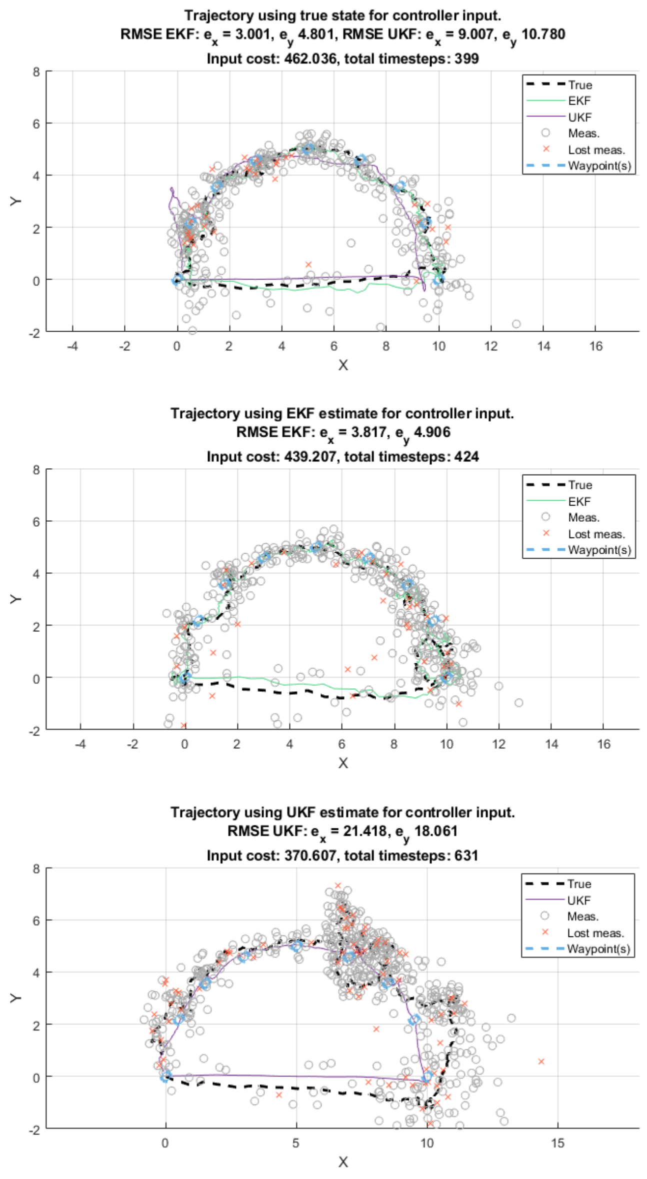

Fig. 15. Simulation of a semi circle trajectory for all controller permutations with $p_{loss} = 10\%$

An aggregate of all the performance metrics for the simulation data for multiple seeds can be seen in Figure 16. The seeds for these simulation runs were picked from `random.org`. It is easy to see that the EKF is much more predictable over different seeds with a lower variance between them. Additionally the EKF had relatively steady RMSE between the true state and the estimates as p_{loss} increases. The only metric where UKF was better was the quadratic input cost, but was still almost twice as slow to reach the final destination, which oddly agrees with the results seen in the initial basic simulations shown in Table II.

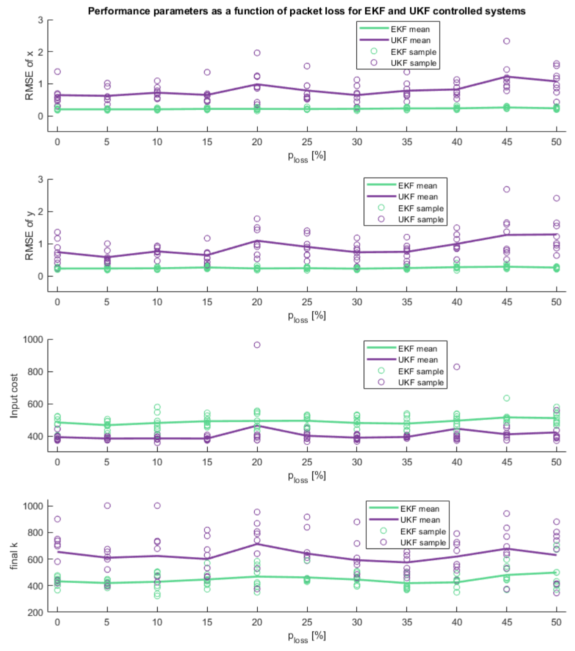

Fig. 16. Performance metrics of the EKF and UKF controlled systems for the semi circle waypoint trajectory (26). The samples are different seeds and the line is their mean.

V. Summary and Discussion

In this paper I have presented the project work for the ELEC-E8105 course. I explained the throught process behind picking a suitable non-linear model, designed an LQR controller with Extended and Unscented Kalman filter for state estimation. The focus of the project work was to look into the effects of measurement quantization and packet loss on the state estimation and control cost.

A. Quantization

I believe my approach to the quantization was probably not the best. Uniform distribution for the non-linear measurement equation proved hard to work with and was the main source of problems for my simulations. Originally, I was looking into producing 3D-plots of quantizer vs. packet loss vs. performance metrics, but due to problems in simulations this didn't prove feasible. Perhaps an adaptive quantization strategy would have proven better, which would be aware of the non-linearities in the measurement equation. Additionally, I do not think my approach on "approximating" the quantization error directly in the innovation covariance calculations is necessarily the best, if not out right the wrong approach.

With my approach, it seems that Unscented Kalman filter outperforms the Extended Kalman filter when heavy quantization is used, however a more diverse set of models and quantization algorithms should probably be used to look into this further. Future ideas could be on properly incorporating the introduced quantization distribution for different types of non-linear filtering models.

B. *Intermittent observations*

Intermittent losses for non-linear filters was the main goal of this project for me. I was rather surprised about how much more robust the Extended Kalman filter is compared to the Unscented variant for this type of system model with packet loss. In networked control systems with a finite number of network channels and more subsystems than communication capacity, one encounters the scheduling problem i.e. which subsystem should be given priority in communicating. The scheduling can be done with something called the cost of information loss, where each subsystem essentially runs their own filters and communication priority is given with the largest error covariances in mind. I had studied this in ELEC-E8123 and was interested in how to expand it for the non-linear case, which was the initial idea for doing this project on the topic, and could be of future interest. The problem with non-linear models becomes the development of the non-linear control system that is robust enough, which by itself could be an entire project topic for another course.

C. MATLAB *code*

The base for my MATLAB code was taken from the courses exercise 4.3, which is under GNU GPL v.2 or later, however the code is heavily modified. I have published my code on `github.com/jk-arp/nlf-proj` should there be any interest in viewing it and I will keep the repository public until the project is marked.

REFERENCES

[1] Simo Särkkä: ”Bayesian Filtering and Smoothing.” Cambridge University Press 2013.

[2] Wikipedia contributors: ”Linear-quadratic regulator” WIkipedia, The Free Encyclopedia, 2020. `Online`, Accessed April 10 2020.

[3] Wikipedia contributors: ”Analog-to-digital converter” WIkipedia, The Free Encyclopedia, 2020. `Online`, Accessed April 10 2020.

[4] Themistoklis Charalambous: ”Basics of sampling and quantization” ELEC-E8123 Course Materials, January 2020.

[5] Themistoklis Charalambous: ”Communication links, data theorems and their effect on control” ELEC-E8123 Course Materials, January 2020.

[6] Wikipedia contributors: ”Quantization (signal processing)” WIkipedia, The Free Encyclopedia, 2020. `Online`, Accessed April 10 2020.

[7] Themistoklis CHaralambous: ”Stabilization of LTI systems over fading channels” ELEC-E8123 Course Materials, February 2020.

[8] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan and S. S. Sastry: ”Kalman filtering with intermittent observations,” in IEEE Transactions on Automatic Control, vol. 49, no. 9, pp. 1453-1464, Sept. 2004.