STATES LAB REPORT
KALMAN FILTER

Elice Roy Kanjamala[1], Giorgi Gelovani[2], Yuxin Zhang[3] Saturday, 12 December 2020

Aim

The aim of this lab is to be familiar with the Kalman Filter, Extended Kalman Filter and Unscented Kalman Filter.

1 Introduction

In statistics and control theory, *Kalman filtering*, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe.

In this lab we are given a DC motor which is driven by an input voltage $u(t)$. We can acquire a measurement $y(t)$ for the angular position of the rotor $\theta(t)$, by an incremental encoder (precision L=512 angles per lap). $\Omega(t)$ denotes the angular velocity ($\Omega(t) = \dot{\theta}(t)$).

To perform a velocity control, we have to estimate online $\theta(t)$ and $\Omega(t)$, from $u(t)$ and $y(t)$.

1.1 Input Voltage

Given that the input voltage $u(t)$ is a zero-mean square wave with period Δ = 100 ms, and a peak-to-peak amplitude A = 0.1 V. This signal is sampled with sample time T_s = 1 ms. We aim to create a MatLab function *u = inputvoltage(D,A,Delta,Ts)*, where u is a column vector which contains the sampled input (square), which provides this sampled input for a duration D=0.5s

We use MATLAB function $square(t)$ to generate a square signal. We require a signal with period 2π and peak-to-peak amplitude 0.1, for the elements of time vector t. Thus, in our case, we generate the input voltage as:

$$u = 0.5 * A * square(\frac{2\pi}{\Delta}t)$$

where t is sampled in $[0, D]$ with a sample time Ts.

1.2 System Modelling and Simulation

We are given a state vector which is represented as:

$$x(t)=\begin{bmatrix}\theta(t)\\ \Omega(t)\end{bmatrix}$$

We know that for a continuous time system, the state space representation is expressed as:

$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases}$$

Therefore, we get a state space representation as:

$$\begin{cases}\dot{x} = \begin{bmatrix}0 & 1\\ 0 & \frac{-1}{T}\end{bmatrix} x + \begin{bmatrix}0\\ \frac{G}{T}\end{bmatrix} u\\ \theta = \begin{bmatrix}1 & 0\end{bmatrix} x + \begin{bmatrix}0\end{bmatrix} u\end{cases}$$

with A=$\begin{bmatrix}0 & 1\\ 0 & \frac{-1}{T}\end{bmatrix}$, B = $\begin{bmatrix}0\\ \frac{G}{T}\end{bmatrix}$, C= $\begin{bmatrix}1 & 0\end{bmatrix}$ and D = $\begin{bmatrix}0\end{bmatrix}$

The input-output signals are sampled with the sample time T_s. The input u(t) is constant between 2 sampling times.

We write a function *[y,x] = simulate(u,G,T,Ts,L,x1)*, where y is a column vector (same size as u). It contains the evolution of the output y_n, x is a 2-columns matrix which contains the evolution of the state vector; x_1 is the initial state vector.

Since our state space representation is in continuous time, we do sampling without any approximation by zero-order-hold method(zoh) using a MATLAB function $c2dm$. For all integer n, and all function f, we note $f_n = f(nT_s)$. Now the discrete time state space is obtained as:

$$\begin{cases}x_{n+1} = \tilde{A}x_n + \tilde{B}u_n\\ y_n = Cx_n + Du\end{cases} \quad with \begin{cases}\tilde{A} = e^{AT_s}\\ \tilde{B} = \int_0^{T_s} e^{A\tau}Bdr\end{cases}$$

The measure y_n provided by the incremental encoder is a quantization of the actual angular position θ_n. It is obtained by:

$$y_n = round(\frac{\theta_n}{2\pi}L)\frac{2\pi}{L}$$

where, $round()$ computes the nearest integer from the given number.

To test simulation , we set G = $50rad.s^{-1}.V^{-1}$ and $T = 20ms$ and call the function $simulate$ with initial value of $x1$= $\begin{bmatrix}0\\ 0\end{bmatrix}$

The input voltage, the angular position of the rotor (simulated and measured) and the angular velocity are shown Figure 1.

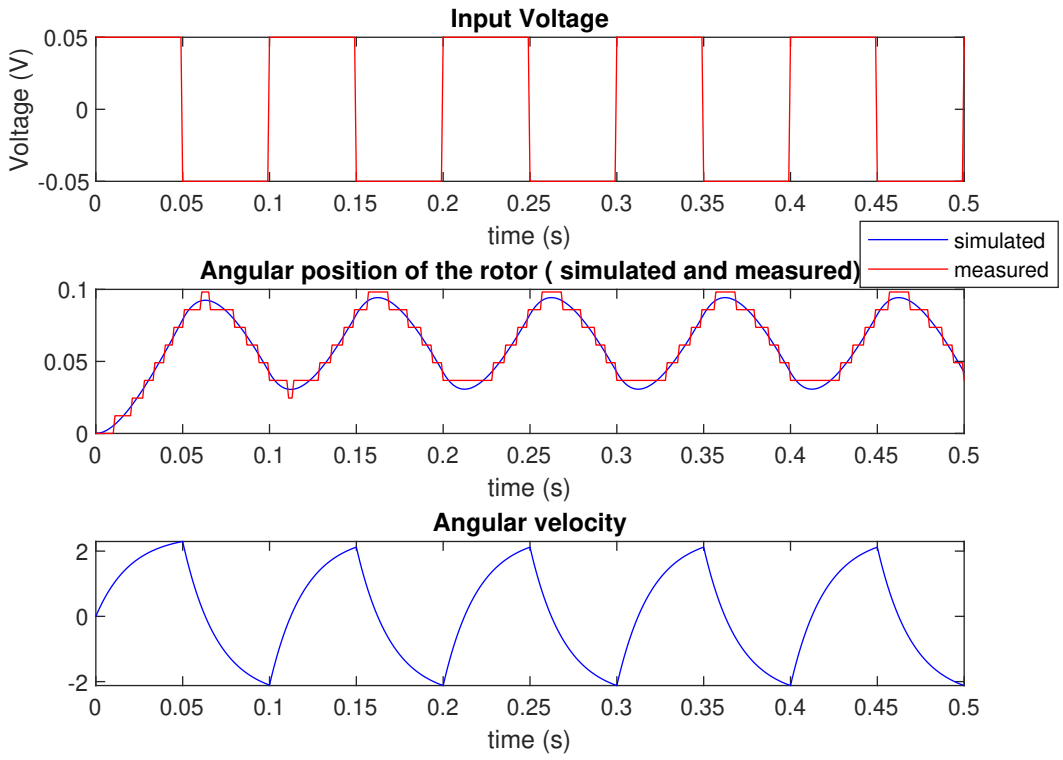


Figure 1: Input voltage, the Angular position of the rotor (simulated and measured) and the Angular velocity

1.3 Kalman Filter

In this part we are going to build the Kalman Filter in order to estimate the state vector X_n. Before we build the model, let's first discuss a bit about what it is and how it is used.

Kalman filter is an optimization algorithm to estimate the state of a system with noise and uncertainties. This filter receives imprecise measures with noise. It is able to estimate current state with good precision and make a prediction of future state. The model of the filter is given below. As we see the equations given below are slightly different from our model's equations, so we are associating them to each other.

$$\begin{cases} x_{n+1} = \tilde{A}x_n + \tilde{B}u_n + \tilde{B}v_n \\ y_n = Cx_n + Du_n + w_n \end{cases} \quad and \quad \begin{cases} x_{n+1} = Fx_n + f_n + V_n \\ y_n = H_n x_n + h_n + W_n \end{cases}$$

By associating above mentioned equations we get:

$$H_n = C, h_n = Du_n, F_n = A, f_n = Bu_n$$

As for the noise in our case w_n is the quantization noise of the incremental encoder, its variance is r ($y_n = \theta_n + w_n$). To take into account modeling errors, we assume that the actual input is $u_n + v_n$, where v_n is a white noise with variance q, independent of w_n. Later we will see that q will be used as a tuning parameter.

$$\text{with } \begin{cases} \mathrm{Y}[n] & = H_n\,\mathrm{X}[n] + h_n + \mathrm{W}[n] \\ \mathrm{X}[n+1] & = F_n\,\mathrm{X}[n] + f_n + \mathrm{V}[n] \end{cases} \quad \text{and} \quad \begin{cases} R_n = C_{\mathrm{W}[n],\mathrm{W}[n]} \\ Q_n = C_{\mathrm{V}[n],\mathrm{V}[n]} \end{cases}$$

Initialization

prediction of $\mathrm{X}[1]$ $\downarrow$	$\hat{\mathrm{X}} \leftarrow m_{\mathrm{X}[1]}$	provides $\hat{\mathrm{X}}^{\vert 0}[1]$
$\downarrow$	$P \leftarrow C_{\mathrm{X}[1],\mathrm{X}[1]}$	$P^{\vert 0}[1]$

Loop $(n \geqslant 1)$

prediction of $\mathrm{Y}[n]$	$\hat{\mathrm{Y}} \leftarrow H_n\,\hat{\mathrm{X}} + h_n$	$\hat{\mathrm{Y}}^{\vert n-1}[n]$
	$C_{\mathrm{X},\mathrm{Y}} \leftarrow P\,H_n^{\mathsf{T}}$	$C_{\mathrm{X},\mathrm{Y}}[n]$
	$C_{\mathrm{Y},\mathrm{Y}} \leftarrow H_n\,P\,H_n^{\mathsf{T}} + R_n$	$C_{\mathrm{Y},\mathrm{Y}}[n]$
$\rightarrow$ observation of $\mathrm{Y}[n]$	$\mathrm{Y} \leftarrow$ sensors	$\mathrm{Y}[n]$
estimation of $\mathrm{X}[n]$	$\hat{\mathrm{X}} \leftarrow \hat{\mathrm{X}} + C_{\mathrm{X},\mathrm{Y}}\,C_{\mathrm{Y},\mathrm{Y}}^{-1}\,(\mathrm{Y} - \hat{\mathrm{Y}})$	$\hat{\mathrm{X}}^{\vert n}[n] \rightarrow$
	$P \leftarrow P - C_{\mathrm{X},\mathrm{Y}}\,C_{\mathrm{Y},\mathrm{Y}}^{-1}\,C_{\mathrm{X},\mathrm{Y}}^{\mathsf{T}}$	$P^{\vert n}[n]$
prediction of $\mathrm{X}[n+1]$ $\downarrow$	$\hat{\mathrm{X}} \leftarrow F_n\,\hat{\mathrm{X}} + f_n$	$\hat{\mathrm{X}}^{\vert n}[n+1]$
$\downarrow$	$P \leftarrow F_n\,P\,F_n^{\mathsf{T}} + Q_n$	$P^{\vert n}[n+1]$

As the motor is initially stopped, we can say that its angular velocity is equal to zero. Whereas the angular position of the rotor θ is measured with an incremental encoder.

The quantization noise can be represented as a variance of a uniformly distributed random variable over an interval of length $2\pi/L$. As it is an uniform distribution and given its variance being $(b-a)^2/12$, we can easily compute the quantization noise $(2\pi/L)^2/12$.

We can initialize θ with any value between $-\pi$ and π and assume the distribution is still uniform. In our case we assume the initial value of θ is 0.

Knowing both angular position and angular velocity, we can build the initial state vector $\hat{X}_{1/0}$.

$$\hat{X}_{1/0} = \begin{bmatrix} \theta \\ 0 \end{bmatrix}$$

After that we can propose a value for its variance $P_{1/0}$. In our case, as the value of angular velocity is equal to zero, besides variance of angular position all will be zero. Thus our covariance matrix will look like the following:

$$P_{1/0} = \begin{bmatrix} (2\,\pi)^2/12 & 0 \\ 0 & 0 \end{bmatrix}$$

As we are done with initialization process we can continue building the model by following the iteration algorithm given above. Finally the output of this function is 2 columns matrix which contains the evaluation of the state vector estimation. Results will be discussed in the final section of the report.

Finally, we build the Stationary Kalman Filter model. The model is given below:

$$\text{with } \begin{cases} \mathrm{Y}[n] &= H\,\mathrm{X}[n] + h_n + \mathrm{W}[n] \\ \mathrm{X}[n+1] &= F\,\mathrm{X}[n] + f_n + \mathrm{V}[n] \end{cases} \quad \text{and} \quad \begin{cases} R = C_{\mathrm{W}[n],\mathrm{W}[n]} \\ Q = C_{\mathrm{V}[n],\mathrm{V}[n]} \end{cases}$$

Preliminaries

Solve/P	$P = F\,P\,F^\intercal - F\,P\,H^\intercal\,(H\,P\,H^\intercal + R)^{-1}\,H\,P\,F^\intercal + Q$	provides $P[\infty]$
Calculate	$K \leftarrow P\,H^\intercal\,(H\,P\,H^\intercal + R)^{-1}$	$K[\infty]$

Initialization

prediction of $\mathrm{X}[1]$ $\downarrow$	$\hat{\mathrm{X}} \leftarrow m_{\mathrm{X}[1]}$	$\hat{\mathrm{X}}^{\lvert 0}[1]$

Loop ($n \geqslant 1$)

prediction of $\mathrm{Y}[n]$	$\hat{\mathrm{Y}} \leftarrow H\,\hat{\mathrm{X}} + h_n$	$\hat{\mathrm{Y}}^{\lvert n-1}[n]$
$\rightarrow$ observation of $\mathrm{Y}[n]$	$\mathrm{Y} \leftarrow$ sensors	$\mathrm{Y}[n]$
estimation of $\mathrm{X}[n]$	$\hat{\mathrm{X}} \leftarrow \hat{\mathrm{X}} + K\,(\mathrm{Y} - \hat{\mathrm{Y}})$	$\hat{\mathrm{X}}^{\lvert n}[n]$ $\rightarrow$
prediction of $\mathrm{X}[n+1]$ $\downarrow$	$\hat{\mathrm{X}} \leftarrow F\,\hat{\mathrm{X}} + f_n$	$\hat{\mathrm{X}}^{\lvert n}[n+1]$

Stationary Kalman filter is highly simplified filter, which we obtained by replacing the time-varying Kalman gain $K[n]$ by its limit $K[\infty]$.

$$K[\infty] = P[\infty]H^T(HP[\infty]H^T + R)^{-1}$$

In Matlab to find the constant gain, we are using $dlqe$ function. This function is Kalman estimator design for discrete-time systems. Given the system:

$$\begin{cases} x[n+1] = Ax[n] + Bu[n] + Gw[n] \\ y[n] = Cx[n] + Du[n] + v[n] \end{cases}$$

with unbiased process noise w[n] and measurement noise v[n] with covariances $E[ww'] = Q$, $E[vv'] = R$, $E[wv'] = 0$.

Associating these equations to our model's equations

$$\begin{cases} x_{n+1} = \tilde{A}x_n + \tilde{B}u_n + \tilde{B}v_n \\ y_n = Cx_n + Du_n + w_n \end{cases}$$

we write the following: $[K,\ ,\ ,\ ] = dlqe(\tilde{A}, \tilde{B}, C, q, r)$;

After defining the Kalman gain, we can write the innovation process which is: $\hat{X}^{\lvert n}[n] = \hat{X}^{\lvert n-1}[n] + K[n](Y[n] - \hat{YX}^{\lvert n-1}[n])$

As for the rest it stays the same, i.e. follows the model given above. This filter asymptotically behaves like the Kalman filter, but it is slower in the transience after the initialization.

2 Simulations

Kalman filter and Sationary Kalman filter are both special cases of Bayes filter whose goal is to write the recursion on the probability distributions which will permit to estimate the current state,

even to predict the future state and the future observation. Kalman filter is a recursive implementation of the MMSE (Minimum Mean Square Estimator) applied to the Gaussian linear model. However, in our DC motor case the initial state $\widehat{\mathrm{X}}^{|0}[1]$ and the sequence w is assumed as uniformly distributed instead of Gaussian. So herein the optimal Kalman filter is an implementation of the LMMSE (Linear MMSE) estimator applying to the linear model.

$$\widehat{x}_{MMSE}(Y) = E(X|Y)$$
$$\widehat{x}_{LMMSE}(Y) = m_X + C_{X,Y}C_{Y,Y}^{-1}(Y - m_Y)$$

The fundamental equation is the state estimation one, which corrects the prediction by means of the Kalman gain $K[n] = C_{X,Y}[n]C_{Y,Y}^{-1}[n]$, the innovation process $\mathrm{Y}[n] - \widehat{\mathrm{Y}}^{|n-1}[n]$ and the error variance $\mathrm{P}^{|n}[n]$.

$$\begin{aligned}\widehat{\mathrm{X}}^{|n}[n] &= \widehat{\mathrm{X}}^{|n-1}[n] + K[n](\mathrm{Y}[n] - \widehat{\mathrm{Y}}^{|n-1}[n]) \\ \mathrm{P}^{|n}[n] &= \mathrm{P}^{|n-1}[n] - K[n]C_{Y,Y}[n]K^T[n]\end{aligned} \tag{1}$$

Stationary Kalman filter is a highly simplified filter (which asymptotically behaves like the Kalman filter) by replacing the time-varying Kalman gain $K[n]$ by its limit $K[\infty]$.

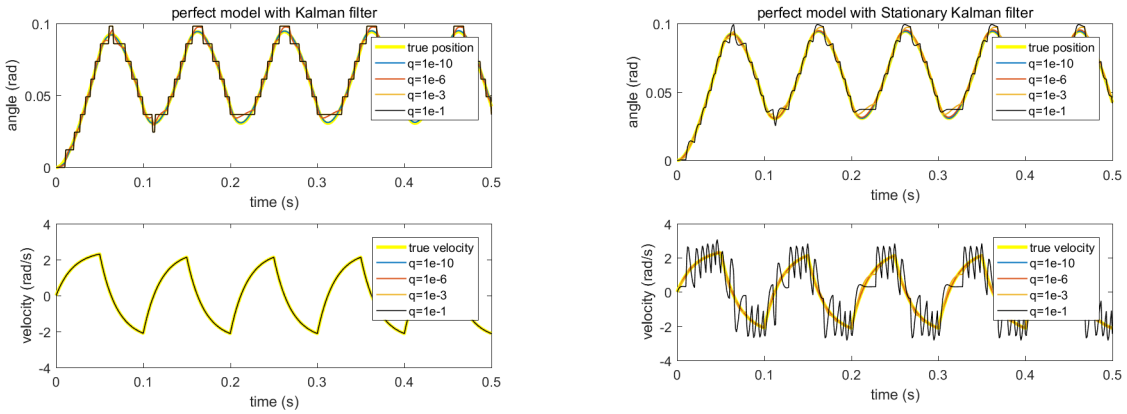
$$K[\infty] = P[\infty]H^T(HP[\infty]H^T + R)^{-1} \tag{2}$$

where, $P[\infty] = FP[\infty]F^T - FP[\infty]H^T(HP[\infty]H^T + R)^{-1}HP[\infty]F^T + Q$.

When we take into account modeling errors, we assumes that the actual input is $u_n + v_n$ instead of only u_n, where v_n is a white noise with variance q. Here we do not know the actual value of q, but we can have the intuition that q is a parameter to tune the trade off between the confidence that you have in the model of the evolution of the state and the confidence you have in the measured angles.

2.1 Use perfect model with $\widehat{\theta}^{|0}[1] = \theta_1$

In this part we assume that our model of the system is perfect ($T_{actual} = T_{filter} = 20ms$), and implement the optimal Kalman filter and the Stationary Kalman filter separately with $q = 1 \times 10^{-10}$, 1×10^{-6}, 1×10^{-3} and 1×10^{-1} respectively. The result is in figure 2 below.



(a) Kalman filter

(b) Stationary Kalman filter

Figure 2: Kalman filter and Stationary Kalman filter applied on the perfect model ($\widehat{\theta}^{|0}[1] = 0$)

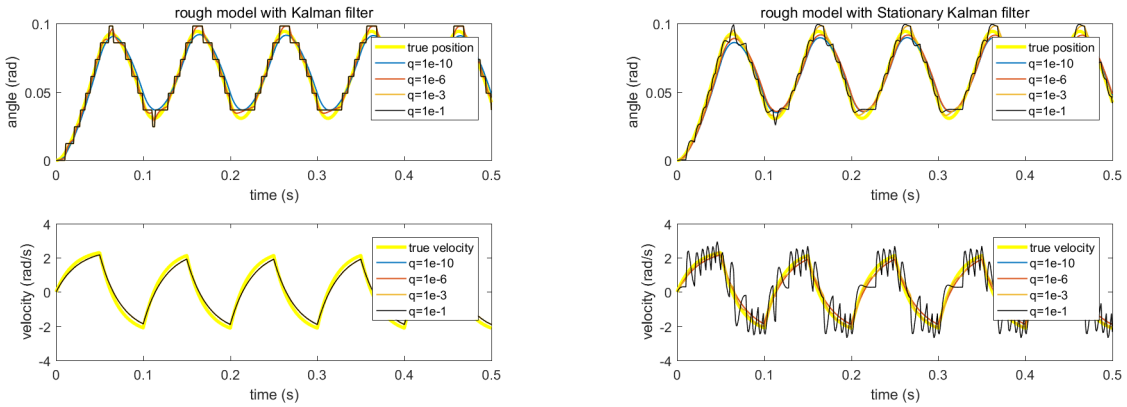
From figure 2, we can say that

- Increasing of value of q keeps the estimated $\widehat{\theta}$ away from the ground true θ but gradually approaches the measured value .
- When we compare the angular position's curves, the Stationary Kalman filter is more likely to believe the model of the evolution of the state than the optimal Kalman filter which is easily to be influenced by the observed values. Because with the same value of q (say 0.1), the estimated $\widehat{\theta}$ is closer to the ground truth with Stationary Kalman filter than with Kalman filter.
- When we apply the Stationary Kalman filter, the estimated value of velocity Ω is sensitive to q. From figure 2(b), we can see that the curve of velocity is very unstable when $q = 0.1$. However, in figure 2(a) with the optimal Kalman filter, q does not affects the measured Ω which is always almost the same as the true velocity.

2.2 Use rough model with $\widehat{\theta}^{|0}[1] = \theta_1$

In the previous part, we implement the Kalman filter and Stationary Kalman filter assuming that the model of the system is perfect. But in practice when we fit some mathematics to this system, the model of the system is never perfect. So maybe the differential equation we have to represent the previous state's model is not a really good model and also the parameters used in the model are not exactly known.

So in this part we change the value of T_{filter} to be $25ms$ ($\neq T_{actual} = 20ms$), and use this rough model to simulate the robustness of the Kalman filter and the Stationary Kalman filter. And the result is in figure 3 below. (Except the value of T_{filter}, the parameters and model used in this part are exactly the same as the last part's.)



(a) Kalman filter

(b) Stationary Kalman filter

Figure 3: Kalman filter and Stationary Kalman filter applied on the rough model ($\widehat{\theta}^{|0}[1] = 0$)

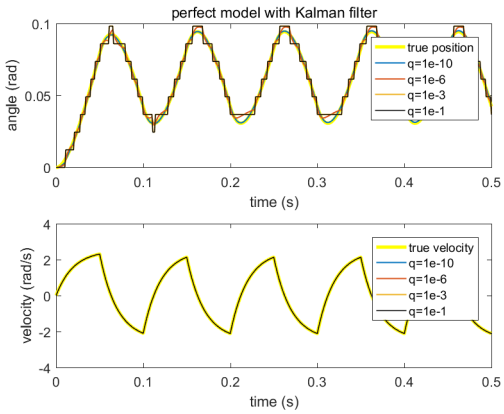
Comments:

- From figure 2(a) to figure 3(a), it is clear that the difference between the estimated velocity and the true velocity becomes larger because the model is not perfect anymore. But the overall performance of the two filters with the rough model does not change much from that with the perfect model.
- Although the two filters' performances of estimating θ are different, they are both not bad. The Stationary Kalman filter is easier than the optimal Kalman filter, but it can also do a good job for the estimation of the angular position.
- We can see that the biggest problem is still the Stationary Kalman filter's sensitivity to q, which leads to a very different velocity value from the true one when q is high (such as $q = 0.1$).

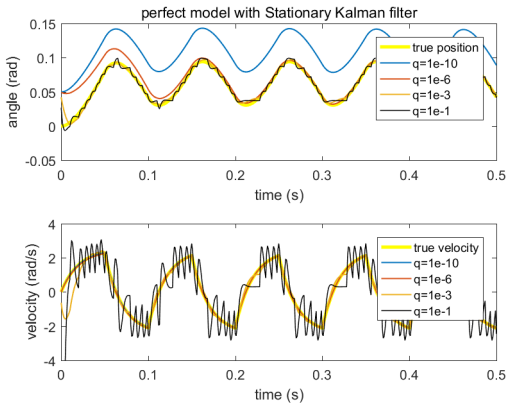
2.3 $\widehat{\theta}^{|0}[1] = \theta_1 \pm 0.05$

In book, it says that the Stationary Kalman filter is slower in the transience after the initialization. So we change the initial value of angular position to visualize this phenomenon. In this part, all of the parameters and model are same as before used in simulation, except the value of $\widehat{\theta}^{|0}[1]$. We repeat the work above with $\widehat{\theta}^{|0}[1] = \theta_1 + 0.05 = 0.05$ and $\widehat{\theta}^{|0}[1] = \theta_1 + 0.05 = -0.05$ respectively.

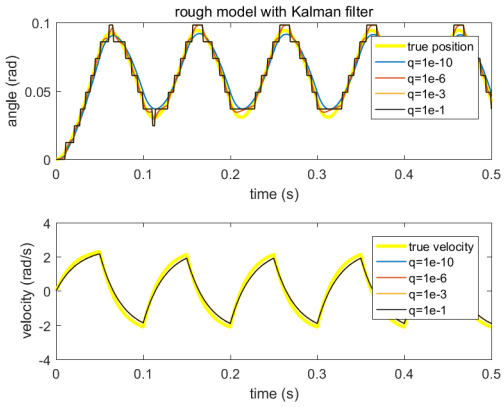
When $\widehat{\theta}^{|0}[1] = 0.05$:



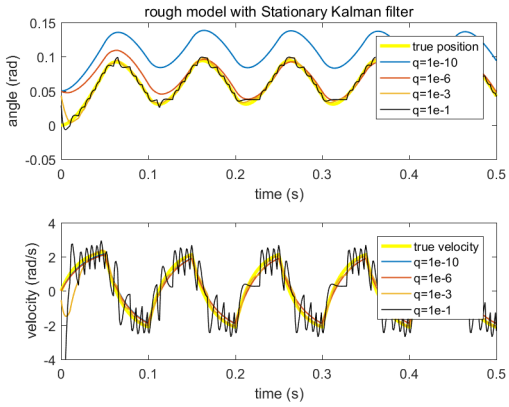
(a) Kalman filter with perfect model



(b) Stationary Kalman filter with perfect model



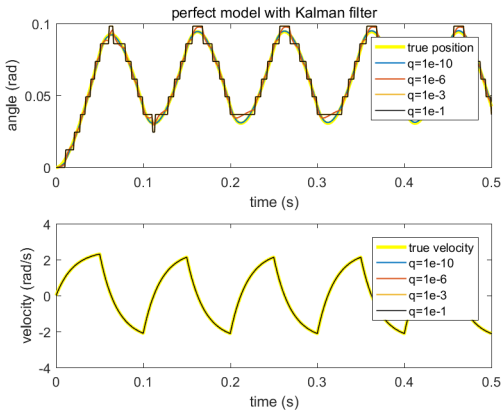
(c) Kalman filter with rough model



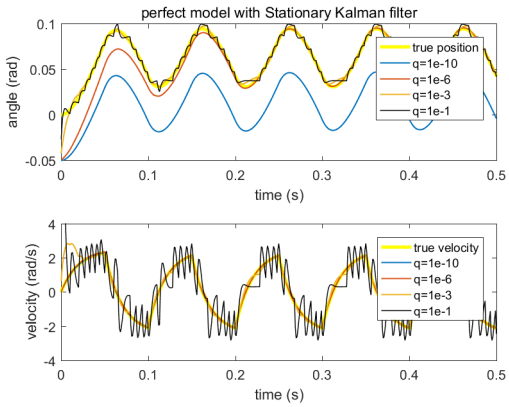
(d) Stationary Kalman filter with rough model

Figure 4: Performances of the two filters when $\widehat{\theta}^{|0}[1] = 0.05$

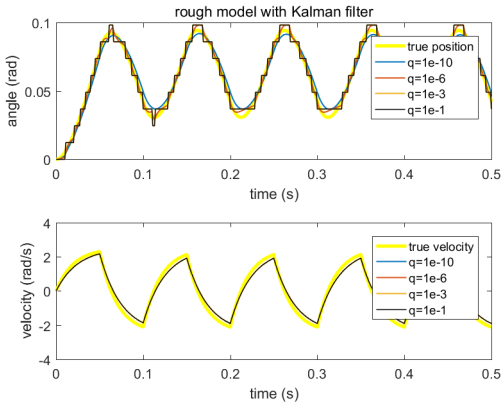
When $\widehat{\theta}^{|0}[1] = -0.05$:



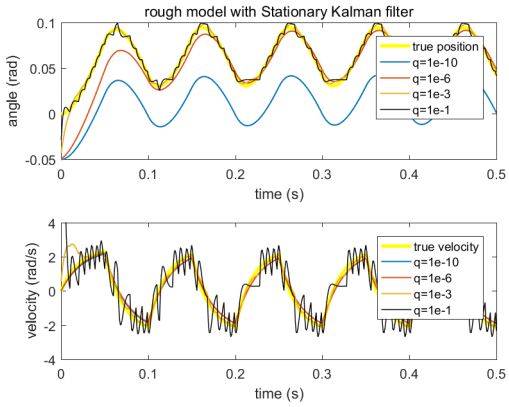
(a) Kalman filter with perfect model



(b) Stationary Kalman filter with perfect model



(c) Kalman filter with rough model



(d) Stationary Kalman filter with rough model

Figure 5: Performances of the two filters when $\widehat{\theta}^{|0}[1] = -0.05$

Discussion about the slower transience:

- From figure 4 and figure 5, changing of the initial value of θ does not influence the optimal Kalman filter much but really affect the Stationary Kalman filter which takes a while to go back to the proper estimation. It's clear to see an obvious deviation at the beginning of not only θ curves but also Ω curves with the Stationary Kalman filter.
- The estimation performance with the optimal Kalman filter is better because there is a very strong $P^{|0}[1]$ with respect to the initial position, so the initial condition is not trusted, and there exists a very strong correction. However the Stationary Kalman filter does not use the initialization of this variance matrix, which results in the difficulty to forget the initial value.

There is another interesting phenomenon with Stationary Kalman filter, which is the overall offset of $\widehat{\theta}$ curves when q is small (say 1×10^{-10}). It reflects that q is a parameter to tune the trade off between the confidence that you have in the model of the evolution of the state and the confidence you have in the measured angles. When q is very small, it means that we do not trust the observation. Conversely, using very large q means that we trust the observation data.

3 Adapting Kalman Filter to Non-Linear Models

In this section we are going to simulate data on a given non-linear model and estimate the state with Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF).

We are given the following model:

$$\begin{cases} \text{x[n+1] = f(x[n],n) + v[n]} \\ \text{y[n] = h(x[n],n) + w[n]} \end{cases}$$

where $v[n]$, $w[n]$, $x[1]$ are normally distributed and zero-mean. We are also given the values for f and h as well as df, dh and the $P1$, Q, R values which are the variances respectively of $x1$, $v[n]$, $w[n]$.

We finalize the initialization by building w and v vectors knowing that they are normally distributed and zero-mean. Finally, we fit the data to the model and draw the results, which are given below.

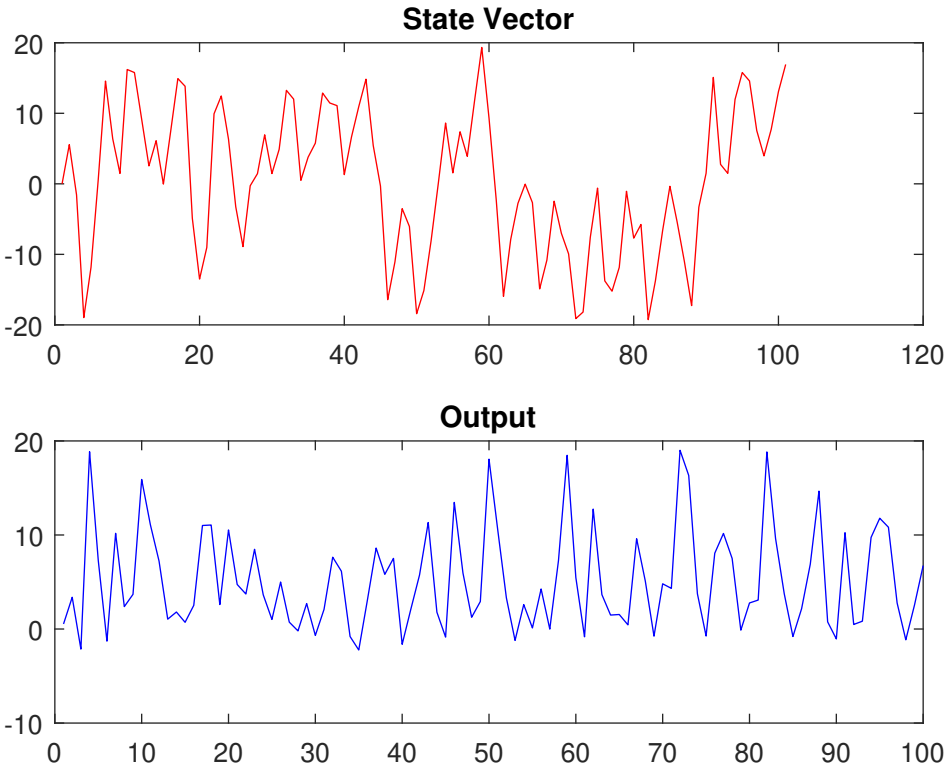


Figure 6: State vector and output vector signals of the given non-linear model

3.1 Extended Kalman Filter

In this subsection we are going to estimate the state with the use of Extended Kalman Filter.

As we know most real world problems involve non linear functions. If we feed a Gaussian with a linear function then the output is also a Gaussian but if we feed it with a non linear function then the output is not a Gaussian. So if we apply a non linear function we get a distribution on which we cannot apply Kalman Filter anymore. As non linearity destroys the Gaussian and it does not make any sense to compute the mean and variance.

So our plan is to approximate those non linear function to make them linear. This brings us to Extended Kalman Filter. In estimation theory, the extended Kalman filter (EKF) is the non-linear version of the Kalman filter which linearizes about an estimate of the current mean and covariance.

The model of the filter is given below

$$\text{with } \begin{cases} \mathrm{Y}[n] &= h_n(\mathrm{X}[n]) + \mathrm{W}[n] \\ \mathrm{X}[n+1] &= f_n(\mathrm{X}[n]) + \mathrm{V}[n] \end{cases} \quad \text{and} \quad \begin{cases} R_n = C_{\mathrm{W}[n],\mathrm{W}[n]} \\ Q_n = C_{\mathrm{V}[n],\mathrm{V}[n]} \end{cases}$$

Initialization

prediction of $\mathrm{X}[1]$ $\downarrow$	$\hat{\mathrm{X}} \leftarrow m_{\mathrm{X}[1]}$	provides $\hat{\mathrm{X}}^{\vert 0}[1]$
$\downarrow$	$P \leftarrow C_{\mathrm{X}[1],\mathrm{X}[1]}$	$P^{\vert 0}[1]$

Loop $(n \geqslant 1)$

Jacobian matrix of h_n	$H \leftarrow \dfrac{\partial h_n}{\partial x^\intercal}(\hat{\mathrm{X}})$	
prediction of $\mathrm{Y}[n]$	$\hat{\mathrm{Y}} \leftarrow h_n(\hat{\mathrm{X}})$	$\hat{\mathrm{Y}}^{\vert n-1}[n]$
	$C_{\mathrm{X},\mathrm{Y}} \leftarrow P H^\intercal$	$C_{\mathrm{X},\mathrm{Y}}[n]$
	$C_{\mathrm{Y},\mathrm{Y}} \leftarrow H P H^\intercal + R_n$	$C_{\mathrm{Y},\mathrm{Y}}[n]$
$\rightarrow$ observation of $\mathrm{Y}[n]$	$\mathrm{Y} \leftarrow$ sensors	$\mathrm{Y}[n]$
estimation of $\mathrm{X}[n]$	$\hat{\mathrm{X}} \leftarrow \hat{\mathrm{X}} + C_{\mathrm{X},\mathrm{Y}} C_{\mathrm{Y},\mathrm{Y}}^{-1} (\mathrm{Y} - \hat{\mathrm{Y}})$	$\hat{\mathrm{X}}^{\vert n}[n] \rightarrow$
	$P \leftarrow P - C_{\mathrm{X},\mathrm{Y}} C_{\mathrm{Y},\mathrm{Y}}^{-1} C_{\mathrm{X},\mathrm{Y}}^\intercal$	$P^{\vert n}[n]$
Jacobian matrix of f_n	$F \leftarrow \dfrac{\partial f_n}{\partial x^\intercal}(\hat{\mathrm{X}})$	
prediction of $\mathrm{X}[n+1]$ $\downarrow$	$\hat{\mathrm{X}} \leftarrow f_n(\hat{\mathrm{X}})$	$\hat{\mathrm{X}}^{\vert n}[n+1]$
$\downarrow$	$P \leftarrow F P F^\intercal + Q_n$	$P^{\vert n}[n+1]$

We build the filter according to the model and draw the estimated states, which we show below.

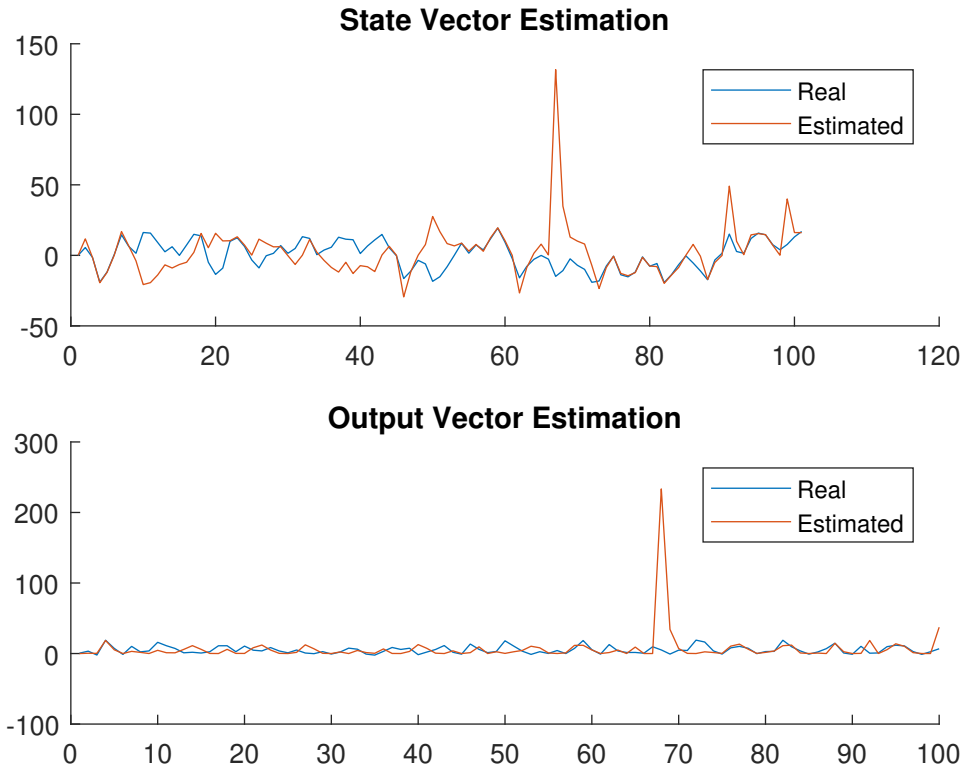


Figure 7: Estimating State with Extended Kalman Filter

3.2 Unscented Kalman Filter

In this subsection we are going to estimate the state with the use of Unscented Kalman Filter.

Unscented Kalman Filter (UKF) is used for better performance. In UKF we have a concept of Sigma Points. We take some points on source Gaussian and map them on target Gaussian after passing through some non linear function and then we calculate the new mean and variance of transformed Gaussian.

It can be very difficult to transform whole state distribution through a non linear function but it is very easy to transform some individual points of the state distribution, these individual points are sigma points. These sigma points are the representatives of whole distribution.

Here the main difference from EKF is that in EKF we take only one point i.e. mean and approximate, but in UKF we take a bunch of points called sigma points and approximate with a fact that more the number of points, more precise our approximation will be!

The model of the filter is given below

$$\begin{cases} x[n+1] = f_n(x[n], v[n]) \\ y[n] = h(x[n]) + w[n] \end{cases} \quad with \begin{cases} R_n = C_{W[n],W[n]} \\ Q_n = C_{V[n],V[n]} \end{cases}$$

Initialization

$$\begin{array}{rcr}
\text{prediction of } \mathrm{x}[1] \downarrow & \hat{\mathrm{x}} \leftarrow m_{\mathrm{x}[1]} & \text{provides } \hat{\mathrm{x}}^{|0}[1] \\
\downarrow & P \leftarrow C_{\mathrm{x}[1],\mathrm{x}[1]} & P^{|0}[1] \\
\text{prediction of } \mathrm{y}[1] \downarrow & (\hat{\mathrm{y}}, C_{\mathrm{y},\mathrm{y}}, C_{\mathrm{x},\mathrm{y}}) \leftarrow \mathrm{UT}(h_1, \hat{\mathrm{x}}, P) & \hat{\mathrm{y}}^{|0}[1], C_{\mathrm{x},\mathrm{y}}[1] \\
\downarrow & C_{\mathrm{y},\mathrm{y}} \leftarrow C_{\mathrm{y},\mathrm{y}} + R_1 & C_{\mathrm{y},\mathrm{y}}[1]
\end{array}$$

Loop $(n \geqslant 1)$

$$\begin{array}{rcr}
\rightarrow \text{observation of } \mathrm{y}[n] & \mathrm{y} \leftarrow \text{sensors} & \mathrm{y}[n] \\
\text{estimation of } \mathrm{x}[n] & \hat{\mathrm{x}} \leftarrow \hat{\mathrm{x}} + C_{\mathrm{x},\mathrm{y}} C_{\mathrm{y},\mathrm{y}}^{-1} (\mathrm{y} - \hat{\mathrm{y}}) & \hat{\mathrm{x}}^{|n}[n] \rightarrow \\
 & P \leftarrow P - C_{\mathrm{x},\mathrm{y}} C_{\mathrm{y},\mathrm{y}}^{-1} C_{\mathrm{x},\mathrm{y}}^{\intercal} & P^{|n}[n] \\
\text{pred. } \mathrm{y}[n+1] \text{ and } \mathrm{x}[n+1] \downarrow & \left(\begin{bmatrix} \hat{\mathrm{x}} \\ \hat{\mathrm{y}} \end{bmatrix}, \begin{bmatrix} P & C_{\mathrm{x},\mathrm{y}} \\ C_{\mathrm{x},\mathrm{y}}^{\intercal} & C_{\mathrm{y},\mathrm{y}} \end{bmatrix} \right) \leftarrow \mathrm{UT}\left(\begin{bmatrix} f_n \\ h_{n+1} \circ f_n \end{bmatrix}, \begin{bmatrix} \hat{\mathrm{x}} \\ 0 \end{bmatrix}, \begin{bmatrix} P & 0 \\ 0 & Q_n \end{bmatrix} \right) & \\
 & & \hat{\mathrm{x}}^{|n}[n+1], \hat{\mathrm{y}}^{|n}[n+1], P^{|n}[n+1], C_{\mathrm{x},\mathrm{y}}[n+1] \\
\downarrow & C_{\mathrm{y},\mathrm{y}} \leftarrow C_{\mathrm{y},\mathrm{y}} + R_{n+1} & C_{\mathrm{y},\mathrm{y}}[n+1]
\end{array}$$

We build the filter according to the model and draw the estimated states, which we show below.

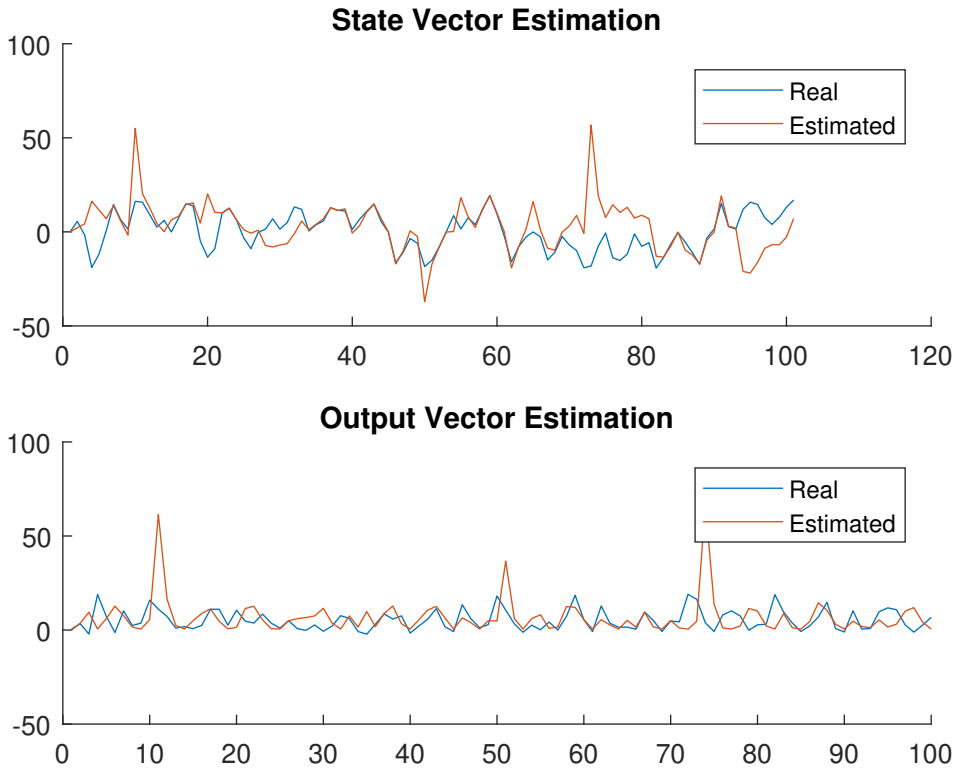


Figure 8: Estimating State with Unscented Kalman Filter

3.3 Conclusion

Both EKF and UKF are some adaptations of the standard Kalman filter to a non linear model. If they are applied to a linear model, we retrieve exactly the Kalman filter.

The UKF seems to provide less estimation error than the EKF. Another advantage is that it does not need to compute the Jacobian matrices.

Code

Listing 1: Matlab Code

```
close all
clear
%% Input voltage
D = 0.5;          % time duration
A = 0.1;          % peak-to-peak amplitude
delta = 100e-3;   % unit: s,  delta=100ms
Ts = 1e-3;        % unit: s,  Ts = 1ms

t = 0: Ts: D;
u = inputvoltage(D, A, delta, Ts);
figure(1), subplot(311), hold off, plot(t,u, 'r')

% System modeling and simulation
G = 50;           % unit: rad/s/V
T = 20e-3;
L = 512;          % with L angles per lap
x1 = [0;0];       % initial state vector [teta, Omega]
[y,x] = simulate(u,G,T,Ts,L,x1);
figure(1), subplot(312), hold off, plot(t,x(:,1), 'b', t, y, 'r' )
figure(1), subplot(313), hold off, plot(t,x(:,2), 'b')

%% 2.3 (b) propose a value of r
r = (2*pi/L)^2/12;            % r = 1.2550e-05

%% 2.3 (c) propose the prior information X_hat 1/0 and its variance P ↩
    1/0 .
x1_0 = zeros(2,1);
x1_0(1) = 0.05;          var_teta = (2*pi)^2/12 ;
x1_0(2) = 0;           var_omega= 0;
P1_0 = [var_teta, 0 ; 0 , 0] ;          % [Cxx, Cxy ; Cyx , Cyy]

%% 2.4 Simulations
q = 0.0001; T = 20e-3;
[ xe ] = kal(y, u, G, T, Ts, L, x1_0, P1_0, q) ;
figure,
plot(t,x(:,1));hold on
plot(t, xe(:,1))

% [ xe ] = skal(y, u, G, T, Ts, L, x1_0, P1_0, q) ;
% figure,
% plot(t,x(:,1));hold on
```

Listing 2: inputvoltage.m

```
function u = inputvoltage(D,A,Delta,Ts)
```

```
% D is duration
% A is peak to peak amplitude
% Delta is period
% Ts is sample time

t=0:Ts:D;
u = (A/2)*square(2*pi*(1/Delta)*t)'; %1/Delta= 10
end
```

Listing 3: simulate.m

```
function [y,x] = simulate(u,G,T,Ts,L,x_new)

%%%%%%%%%%%%%%%%%%Initialization%%%%%%%%%%%%%%%%%%
len_u = length(u);
y = zeros(len_u,1);
x = zeros(len_u,2);

%%%%%%%%%%%%%%%%%%System modeling%%%%%%%%%%%%%%%%%%
A = [0 1;0 -1/T];
B = [0;G/T];
C = [1 0];
D = 0;

[Ad,Bd,~,~] = c2dm(A,B,C,D,Ts,'zoh');  % Discretization


for n = 1:len_u
    x(n, :) = x_new';
    x_new = Ad*x_new  + Bd*u(n);
end

teta = x(:,1);
y = round(teta*L/2/pi) * 2*pi/L;

end
```

Listing 4: Kalman Filter

```
function [ xe ] = kal(y, u, G, T, Ts, L, x1_0, P1_0, q)
A = [0 1;0 -1/T];  B = [0;G/T];  C = [1 0];  D = 0;
[Ad,Bd,~,~] = c2dm(A,B,C,D,Ts,'zoh');

xe = zeros(length(u), 2);

H = C;  h = 0;  r = 1.2550e-05 ; % r is a value need to be proposed
F = Ad;   % we have f in loop and q as an input parameter

%%%%%%%%%%%%%  1st iteration %%%%%%%%%%%%%
```

```
y_hat = H * x1_0 + h ;
Cxy = P1_0 * H' ;
Cyy = H * P1_0 * H' + r ;

x1_1 = x1_0 + Cxy / Cyy * (y(1) - y_hat) ; xe(1,:) = x1_1' ;
P1_1 = P1_0 - Cxy / Cyy * Cxy';

f = Bd * u(1);
x = F * x1_1 + f ;
P = F * P1_1 * F' + q ;
%%%%%%%%%%%%% n > 1 iteration %%%%%%%%%%%%
for n = 2 : length(u)
    y_hat = H * x + h ;
    Cxy = P * H' ;
    Cyy = H * P * H' + r ;

    x = x + Cxy / Cyy * (y(n) - y_hat) ;      xe(n,:) = x' ;
    P = P - Cxy / Cyy * Cxy';

    f = Bd * u(n);
    x = F * x + f ;
    P = F * P * F' + q ;
end

end
```

Listing 5: Stationary Kalman Filter

```
function [ xe ] = skal(y, u, G, T, Ts, L, x1_0, P1_0, q)
A = [0 1;0 -1/T];  B = [0;G/T];  C = [1 0];  D = 0;
[Ad,Bd,~,~] = c2dm(A,B,C,D,Ts,'zoh');

xe = zeros(length(u), 2);

H = C;   h = 0;   r = (2*pi/L)^2/12; % r = 1.2550e-05 ; % r is a value need to be proposed
F = Ad;                                % we have f in loop and q as an input parameter

[K,~,~,~] = dlqe(Ad, Bd, C, q, r) ; % Kalman gain

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y_hat = H * x1_0 + h ;

x1_1 = x1_0 + K * (y(1) - y_hat) ;       xe(1,:) = x1_1 ;
f = Bd * u(1);
x = F * x1_1 + f ;


for n = 2 : length(u)
```

```
21      y_hat = H * x + h ;
22
23      x = x + K * (y(n) - y_hat) ;       xe(n,:) = x ;
24      f = Bd * u(n);
25      x = F * x + f ;
26  end
27  end
```