Ad Hoc and Sensor Networks: Kalman Filter Report

Rodrigo Tamaki

November 2024

1 Introduction

As a brief introduction, the Kalman Filter [1] is a set of equations that recursively allow us to estimate the state of a certain process. The state (at an instant k), denoted as x_k, can be expressed as the linear combination of the previous state x_{k-1} with other elements such as noise w_k and an optional control input u_{k-1} (*see expression* (1)):

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

However, the sensors that we might deploy do not directly obtain these x_k values but rather some measurements z_k, characterized by expression (2):

$$z_k = Hx_k + v_k \tag{2}$$

Here, v_k represents the noise interfering with the measurement. Note that both noises (in the process and in the measurement) are assumed to be Gaussian, with covariance matrices denoted as Q and R, respectively.

The previous state x_{k-1} and the measurement z_k allow us to calculate the *a priori* and *a posteriori* estimates of the current state x_k while minimizing the estimation error. After reordering some of the expressions shown in [1], we arrive at expression (3), which represents the purpose of the filter:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{3}$$

Basically, this equation describes that, in order to obtain the estimate of the real state, we need the *a priori* estimate of the current state and a weighted difference, known as the "measurement innovation." This difference is the discrepancy between the measured value z_k and the estimate $H\hat{x}_k^-$, with its importance determined by the coefficient K, also called the "Kalman gain" or "blending factor." This estimation process is carried out recursively in two different stages (*see Figure 1*):

- **Time Update:** Set of equations that calculates the *a priori* estimate of the following time instant from the current state estimate and the covariance error.
- **Measurement Update:** Set of equations in charge of the feedback process. These improve the *a posteriori* estimate by incorporating new measurements into the *a priori* estimate.

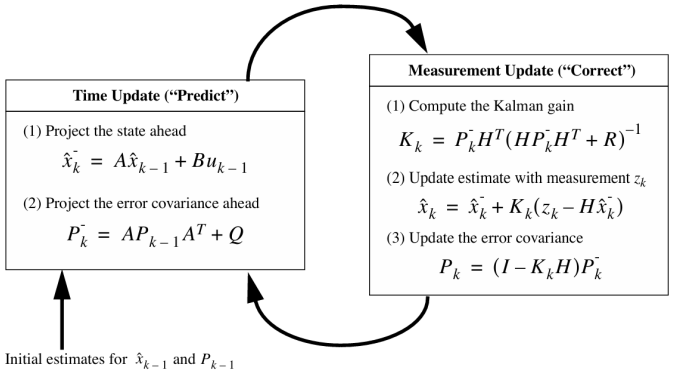

Figure 1: Kalman Filter Algorithm Scheme

Depending on the conditions, this algorithm can provide a reliable estimate of the true value of the process state once the estimation of the error covariance and the Kalman gain stabilizes.

2 Kalman Filter Implementation

For the implementation of the Kalman Filter, I decided to use MATLAB since it is highly convenient for Telecommunications and Signal Processing tasks. In this section, I explain the development of the code and present the simulation settings and results.

This report considers the regular Kalman Filter, not the Extended Kalman Filter (EKF), as the estimated process is completely linear. The MATLAB code used for the implementation is shown below. The source code is also available on GitHub at `https://github.com/rodritamaki/Kalman-Filter-Report.git`.

```
% Input Parameters
true_x = randn;
Q = 1e-5;
R_values = [0.001, 0.01, 0.1, 1];
num_meas = 50;

% Generate random measurements (same across all R for consistency)
z = true_x + sqrt(R_values(2)) * randn(1, num_meas);

% Arrays to store results
all_estimates = zeros(numel(R_values), num_meas);
all_covariances = zeros(numel(R_values), num_meas);

% Kalman Filter for different R values
for r_idx = 1:numel(R_values)
    R = R_values(r_idx);
    sigma_R = sqrt(R);

    % Initial conditions
    x_hat = 0;
    P = 1;

    % Arrays for storing the estimated values
    x_estimates = zeros(1, num_meas);
    P_values = zeros(1, num_meas);

    % Kalman Filter recursive algorithm
    for k = 1:num_meas
        % Time Update Equations (Predict)
        x_hat_prior = x_hat;
        P_prior = P + Q;

        % Measurement Update Equations (Correct)
        K = P_prior / (P_prior + R);
        x_hat = x_hat_prior + K * (z(k) - x_hat_prior);
        P = (1 - K) * P_prior;

        % Results storage
        x_estimates(k) = x_hat;
        P_values(k) = P;
    end

    % Store results for this R
    all_estimates(r_idx, :) = x_estimates;
    all_covariances(r_idx, :) = P_values;
end
```

Figure 2: MATLAB implementation of the Kalman Filter to estimate a random constant. This code evaluates the filter's performance for varying measurement noise covariance (R).

2.1 Code Explanation

As shown in *Figure 2*, the code is structured in a straightforward manner, with two main sections.

The first section involves the definition of the variables, simulating the *off-line* process. At the top, we generate the random constant using the **randn** function (based on the normal distribution) and allow the user to customize key parameters, such as the covariance of the noise and the number of measurements. After defining the measurement noise covariance (R), we calculate the measurement values according to expression (2). It is important to note that this implementation follows the methodology described in [1], and the structure of the experiments is entirely based on it. Generating the measurements prior to running the experiments ensures that the same measured values are used across all tests. This approach allows us to isolate and evaluate the impact of the parameter of interest (in this example, R) under consistent conditions [1].

The second section (the `for` loop) contains the Kalman Filter algorithm itself. Initially, the conditions are defined. Since the true value is generated from the normal distribution, I decided to initialize the estimate with the mean value, which is zero, and the covariance of the normal distribution. The implementation of the algorithm follows the standard prediction and correction process, as described in *Figure 1*. Some simplifications are evident in the expressions due to the following assumptions:

- The state does not change during the process, which implies $A = 1$.
- There is no control input, meaning $u_k = 0$.
- The measurement is a direct observation of the state, so $H = 1$.

Finally, all estimated values are stored for future graphical representation and analysis.

2.2 Simulation Settings

One of the main objectives of this work, as outlined in the guidelines, is to study the impact of the measurement noise covariance (R) on the performance of the Kalman Filter. Thus, the first scenario, referred to in this report as *Case Study 1*, involves running the algorithm for four different values of R (0.001, 0.01, 0.1, and 1). Consistent with the setup described in [1], the number of measurements has been fixed at 50, with a process noise covariance of $Q = 1e - 5$.

It is worth noting that the measurements were generated using $R = 0.01$ (*see line 8 in Figure 2*). This suggests that the best performance is likely to be achieved when the filter is configured with this corresponding value of R.

In *Case Study 2*, the experiment explores the effect of varying the process noise covariance (Q) with values $1e-7$, $1e-5$, and $1e-3$. This is an important parameter to investigate, as Q influences the variability of the real states and directly impacts the time update section of the Kalman Filter algorithm. For consistency, the measurement noise covariance (R) is fixed at 0.01 across all three cases.

Finally, *Case Study 3* investigates the effect of the initial value of the error covariance (P_0) on the filter's convergence. As hypothesized, the convergence behavior of the estimation is expected to depend significantly on this initial value. According to [1], if there is uncertainty regarding the correctness of the chosen initial estimate ($\hat{x}_0$), setting $P_0 = 0$ should be avoided, as it will prevent the algorithm from converging. Nevertheless, this scenario is included in the experiments to observe its effect.

2.3 Results

2.3.1 Case Study 1: Impact of R Value

The measurement noise covariance (R) plays a critical role in the Kalman Filter, as it determines the balance between trusting the *a priori* estimate ($\hat{x}_k^-$) and the measurements (z_k) (*see Figure 1*). Intuitively, a smaller R indicates greater trust in the measurements, while a larger R places more emphasis on the prior estimate.

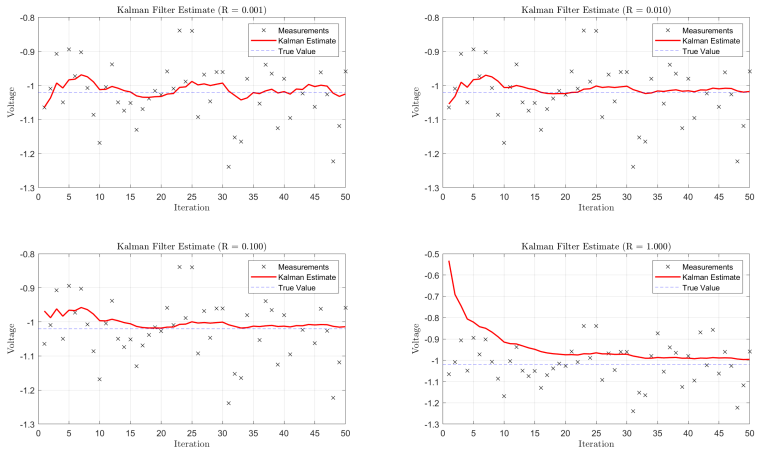

Figure 3: First Simulation: $R = (0.001, 0.01, 0.1, 1)$. The true value of x (-1.0203) is represented by the dashed line, the measurements by the cross marks, and the filter estimate by the red curve.

If $R = 0$, the Kalman Gain (K) reaches its maximum value of 1, meaning the filter completely trusts the measurements. Conversely, as R increases, the value of K decreases, indicating a greater reliance on the prior estimate over the measurements. This trade-off directly affects the filter's performance and convergence behavior.

Figure 3 illustrates the performance of the filter for the selected R values, where the behavior aligns with the theoretical expectations:

- Best Performance: For $R = 0.01$, the filter achieves an optimal balance between trusting the prior estimates and the measurements. This results in smooth convergence to the true value with minimal fluctuations.
- Small R (e.g., $R = 0.001$): The filter converges but exhibits significant fluctuations due to its high sensitivity to measurement noise, as the small R value gives the measurements excessive weight.
- Large R (e.g., $R = 1$): The filter relies heavily on the prior estimates, especially in the initial iterations. While this reduces abrupt changes caused by measurement noise, the filter does not converge to the true value, even after 50 iterations. This convergence issue can be observed in *Figure 4* as well.

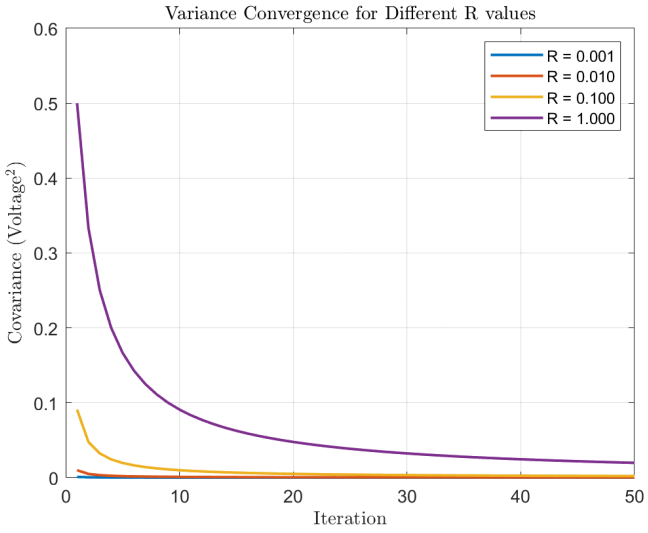

Figure 4: Evolution of error covariance P_k^- for each R value.

In practical scenarios, choosing the right R value depends on the nature of the measurements. If the measurements are noisy or their accuracy is uncertain, a larger R may be appropriate, especially in cases where there are fewer

restrictions on the number of iterations. However, if the measurements are reliable, a smaller R can enable faster and more precise convergence, provided the measurement noise does not dominate the process.

2.3.2 Case Study 2: Impact of Q Value

The process noise covariance (Q) allows the Kalman Filter to account for uncertainty in the state estimation caused by process noise. This parameter adjusts how the algorithm updates the error covariance during the prediction step. Since the actual nature of the process noise is typically unknown (though assumed Gaussian), selecting an appropriate approximation for Q is crucial for balancing responsiveness and stability.

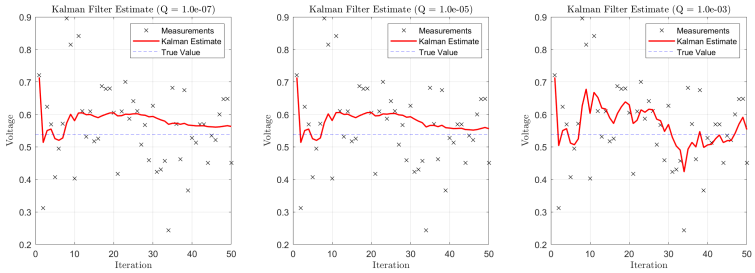

Figure 5: Second Simulation: $Q = (1e-7,\ 1e-5,\ 1e-3)$. The true value of x is 0.5377, represented by the dashed line.

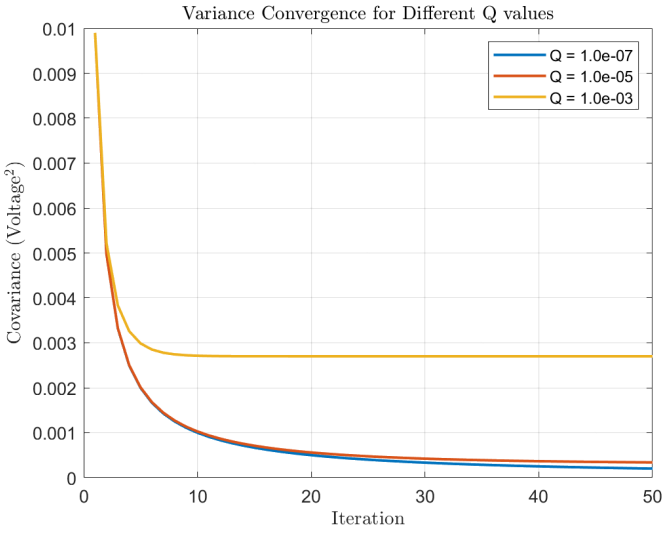

Figure 6: Evolution of error covariance P_k^- for each Q value.

The results in *Figure 5* demonstrate the influence of different Q values on the filter's performance:

- $Q = 1e-3$: The filter assumes significant process uncertainty, causing it to react faster to changes in measurements. This results in quicker convergence, but at the cost of a noisier estimate.
- $Q = 1e-5$: This value provides a good balance, with a reasonably fast response and a smoother estimate, converging effectively to the true value within the given iterations.
- $Q = 1e-7$: The filter assumes minimal process noise, leading to a smoother but slower response. Although the estimate improves gradually, it does not reach the true value within the 50 iterations displayed. Based on the trend, it is likely to converge eventually but would require significantly more iterations.

The evolution of the error covariance (P_k^-) in *Figure 6* further supports these observations. Higher Q values cause the covariance to decrease more rapidly, indicating quicker stabilization of the filter.

After conducting this experiment, I concluded that this approach may not be the most appropriate for evaluating the impact of Q in this specific scenario. Since the Kalman Filter is estimating a constant value that does not change over time, there is no inherent process noise in this case. As a result, the parameter Q does not directly reflect its typical role in dynamic systems.

In future experiments, it might be more insightful to evaluate the impact of Q on a system with actual state variability, such as a tracking or prediction scenario where the state changes or fluctuates over time. This would better demonstrate the parameter's ability to balance responsiveness and stability in dynamic processes.

2.3.3 Case Study 3: Impact of P_0

The initial value of P represents the level of confidence placed in the initial state estimate ($\hat{x}_0$). As shown in *Figure 1*, setting $P = 0$ results in a Kalman Gain K of zero, meaning the filter entirely disregards the measurements. This, combined with the low value of Q used in this experiment, causes the convergence to be excessively slow. It is important to note that this scenario assumes the initial estimate ($\hat{x}_0$) does not exactly match the true value.

For the other initial values of P, the filter's performance is nearly identical. The error covariance converges early in the process, as illustrated in *Figure 8*. By the 20^{th} iteration, the covariance has stabilized and reached the same value for all three cases, indicating that the choice of initial P has little impact on the filter's long-term performance when the initial estimate is reasonably accurate.

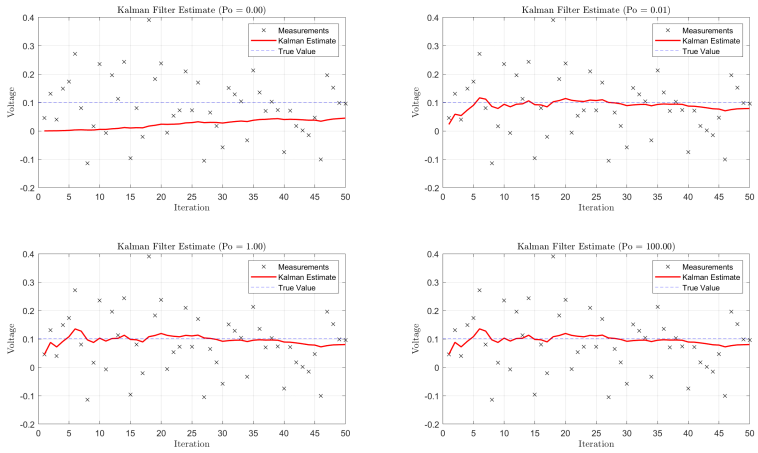

Figure 7: Third Simulation: $P_0 = (10,\ 0.01,\ 1\ and\ 100)$. The true value of x is 0.1001, represented by the dashed line.

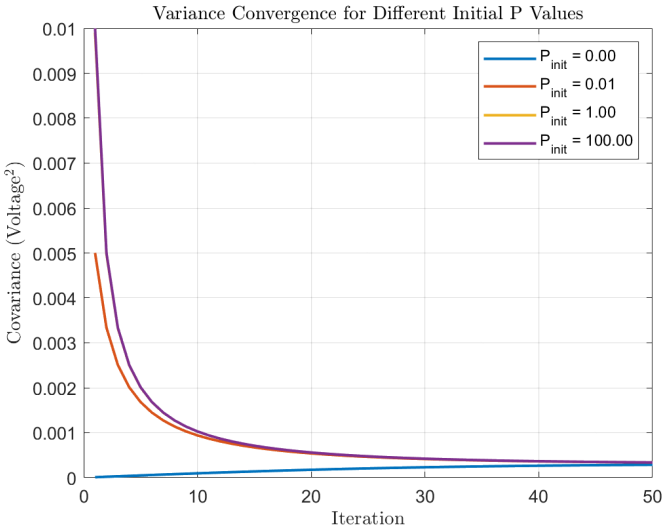

Figure 8: Evolution of error covariance P_k^- for each P_0 value.

3 Conclusion

The evaluated parameters have clearly noticeable impacts on the performance of the filter. At the same time, I believe that selecting the most appropriate values can be a challenging task, as many processes may be completely uncharacter-

ized, leaving us unable to begin with coherent estimates of certain statistics. As illustrated in [1], this selection process is often performed off-line with the assistance of another Kalman Filter.

Even in the simple case of estimating a random constant, we observed interesting behaviors. Regarding R , it has been verified that it enables careful control over the smoothness of the estimation by balancing the weights given to the measurements and predictions. Q introduces the concept of process uncertainty into the algorithm, allowing it to account for potential errors in the system's dynamics. This parameter showed a significant impact on the filter's responsiveness and stability. In particular, higher values of Q resulted in faster convergence but noisier estimates, while smaller values provided smoother but slower responses. In the studied case, there was no process noise, leaving the task of evaluating the performance under the presence of noise and, further, time-varying noise for future exploration.

Finally, the initial estimate of P_0 has been shown to be a non-critical parameter for long-term performance, as it had little effect on the results once the filter stabilized. However, it is important to avoid using an initial value of zero in cases where the initial estimate is uncertain, which I would say is common in most scenarios. Setting $P_0 = 0$ prevents the filter from properly updating its state, leading to a lack of convergence.

References

[1] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," 2006. [Online]. Available: `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`