Exercises in Kalman filtering

isaed275

Non-linear time series analysis

1 Background and objectives

The objective of these exercises is to write and implement Kalman filters for filtering of two systems of different dimensions and complexity.

The Kalman filter is a mathematical algorithm that is used to estimate the state of a system based on noisy measurements. The Kalman filter works by making an estimation based on the modeled state and covariances for the different kinds of noise, and then recursively updating its prediction based on previous performance.

In the prediction step, the update equation of the state is first evaluated like

$$\hat{x}_{n+1} = Ax + \omega_n \tag{1}$$

The error covariance is also projected ahead, according to

$$P_n^- = AP_{n-1}A^T + Q_\nu \tag{2}$$

The filter then updates the state estimate based on the measurement received at the current time step.

$$G_n = \frac{P_n^- B^T}{BP_n^- B^T + Q_\nu} \tag{3}$$

After the gain is calculated, the state is updated, using the known value from the measurement y_m to get the prediction error.

$$\hat{x}_n = \hat{x}_{n-1} + G_n(y_{m_n} - B\hat{x}_{n-1}) \tag{4}$$

Lastly, the error covariances are updated using the same measurement and gain

$$P_n = P_n^-(I - GB) \tag{5}$$

This chain of prediction and updating is then repeated for each time step. The Kalman filter calculates an optimal estimate of the state by minimizing the mean squared error between the predicted and measured values.

1.1 Constant voltage

We have a voltage held constant with no input, but with minor process noise. The state update model can be formulated as

$$x_{n+1} = x_n + \omega_n \tag{6}$$

e.g. $A = B = 1$, since there is no inherent dynamics and one sole state variable. We have a measurement such that

$$y_n = x_n + \nu_n \tag{7}$$

where ν_n is the measurement noise with the covariance $Q_\nu = Q_0 = 0.1^2 V^2$

1.2 Harmonic oscillator

We have a harmonic oscillator which is described by the differential equation

$$\frac{d^2y(t)}{dt^2} + \omega^2 y(t) = 0 \tag{8}$$

where $\omega = 2\pi$. We now want to describe these dynamics to a two-dimensional first order state space model. By rewriting Equation 8 as

$$\ddot{y}(t) = -\omega^2 y(t) \tag{9}$$

it is now easy to see the possible realization of the state variables

$$x_1 = y(t) \tag{10}$$
$$x_2 = \dot{y}(t) \tag{11}$$

which together with Equation 9 gives the full state space model

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\omega^2 x_1 \end{aligned} \tag{12}$$

Using Euler's numerical approximation, we can now write it as a linear system of the form

$$x_{n+1} = Ax_n + \omega_n \tag{13}$$

where $\omega_n =$ is Gaussian noise with the standard deviation $\sigma_\omega = 0.01$, and A are the coefficients in the state space model in Equation 12

$$A = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \tag{14}$$

which makes up the transition matrix.

In the implementation of the measurement, the measured variable is the position of the oscillator, which in conjunction with the state in Equation 11 gives us the measurement matrix

$$B = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{15}$$

There is also a known measurement noise in the provided time series. This noise, ν_n has a standard deviation $\sigma_\nu \approx 0.25$ which gives us the full measurement model

$$y_n = Bx_n + \nu_n \tag{16}$$

2 Methodology

2.1 Kalman filter for measurement data of a constant voltage

For the measurement, a series of 50 normally distributed values with $\sigma = 0.1$ were generated with a set seed using MATLAB's function *randn*. This series was added on to the known true value of the voltage to create the measurement data.

Thanks to the fact that MATLAB code is relatively close to mathematical notation, the implementation of the Kalman filter is very similar to equations 1 - 5. However, becuase of the simple nature of this specific model, some simplifications could be made. The previously described steps of the Kalman filter operation could thus be simplified as

```
x_pred = x; % No change predicted since constant
    dynamics
P_pred = P + Q_w; % Update for the prediction-error
    covariance

G = P_pred / (P_pred + Q_ny); % Calculating Kalman
    gain
x = x_pred + G * (y(i) - x_pred); % Update based on
    certainty
P = (1 - G) * P_pred;
```

With the fact that the model for the voltage is static, and that there is only one state variable, the state parameters A and B are both equal to 1. The full code, excluding graphics generation can be found in Appendix A. As the Kalman filter in its nature is recursive, two initial guesses for the state and error covariance had to be made. Given enough data points, the initial guesses $x_0 = 0.5$ and $P_0 = 0.1$ should not be of great importance as the filter will correct itself. However, with only 50 measured points there was a small risk that a too high or low initial value could show lasting errors in the measurement.

2.2 Kalman filter for the two-dimensional model of a harmonic oscillator

For the two-dimensional case of the harmonic oscillator, the same process was followed, but this time utilizing the full model described in section 1. The main difference besides an expanded state model was that the covariance matrices were unknown. To find the process noise covariance, two surrogate time series with equivalent dynamics were generated. One pure and one corrupted with dynamic noise with known standard deviation $\sigma_\omega = 0.01$.

```
function dydt = noisy_harmonic_oscillator(t, y)
    sigma_omega = 0.01;
    noise = sigma_omega * randn(size(y));
    w = 2*pi;
    dydt = [y(2); -w^2*y(1)] + noise;
end
```

The difference was taken from the surrogates and fed into MATLAB's function *cov*.

The measurement error covariance was calculated simply as a scalar equal to the square of the given standard deviation. This because of the assumption that the measurement noise is uncorrelated to any specific state variable.

3 Results

In Figure 1, we see the measurements filtered using the given covariances, which should according to theory be mathematically optimal. The estimated voltage oscillates a bit around the known true value but eventually settles very close to the true value. In the lower plot, we see the progression in time of the filtering-error covariance.

When the covariance of the measurement noise, Q_0, is multiplied by 100, the initial state estimation persists for longer and the estimated voltage is not as responsive to changes in the meaurement as in the original case. With Q_0 set to 1/100 of its initial value, the estimated voltage follows the measurement closely and the filter's effect can be considered weaker. The higher filter-error covariance indicates that the degree of uncertainty is higher in this case.

The estimated position of the harmonic oscillator, visible in Figure 4 shows behaviour reminiscent of that in Figure 3, where the filtered position follows the measurement closely.

4 Discussion

The reason for the "jumpy" behaviour in the case of the lower Q_0 can be seen with a combination of Equations 3 and 4. When the covariance of the measurement noise decreases, the Kalman gain increases, which increases the measurement's impact in the state update equation. This leads to a dynamic that

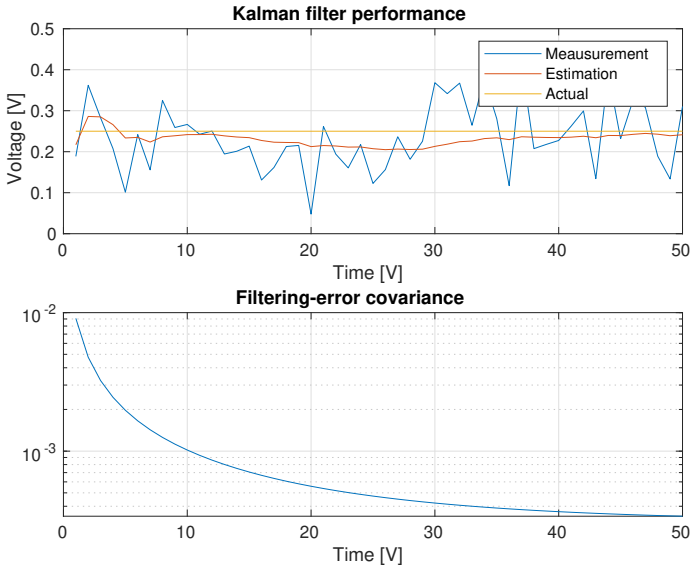

Figure 1: $Q_\nu = Q_0$

follows the measurements closely. The covariance matrices can be seen as "doubt parameters" towards the measurement versus the model. If the ratio of covariance of model error to measurement error increases, one can visualize it as if we distrust the model, leading to a "slow" filter. Conversely, we can see in Figure 2 that a higher Q_v leads to the filter staying closer to the model.

This brings us to the result from the harmonic oscillator, visible in Figure 4. While the filtered curve is smoother than the measured, the result looks unimpressive and the filtered curve is jagged and follows the measurement noise closely. There are several possible reasons for this.

One reason may be that the covariance matrix is incorrectly calculated, perhaps because of an unidentified error in the method of surrogate data comparison for the covariance extraction. However, some tuning after the fact was made, where the error covariance matrices were increased, decreased and restructured. The visual image of the plot and the filter-error covariance at the last time step were analyzed to see what impact different tuning setups would give. These tries showed that while there was room for variability, no configuration produced a curve smooth enough to resemble a pure sinusoid and most seemingly reasonable ratios of covariances led to the filter-error covariance converging to very similar values. This leaves us with the two obvious hypotheses for explaining this behaviour. One of them being a subtle error in the implementation of the filter, and the other being that the level of noise in the data set is too high to facilitate a clean signal, even after filtering.

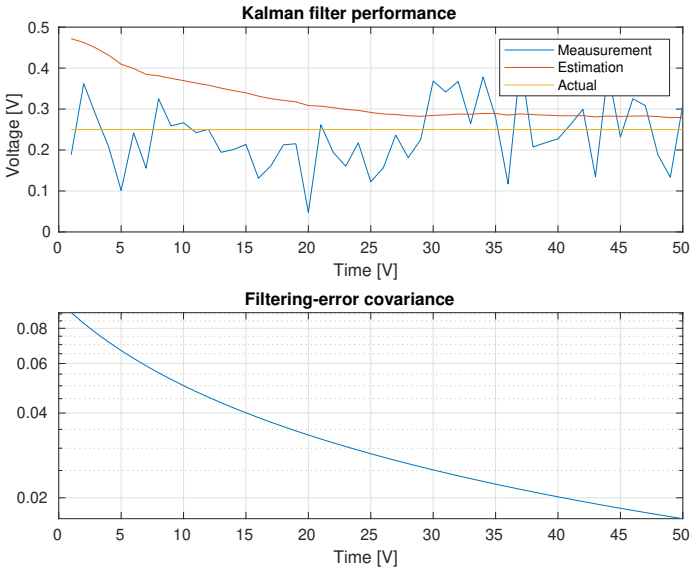

Figure 2: $Q_\nu = 100Q_0$

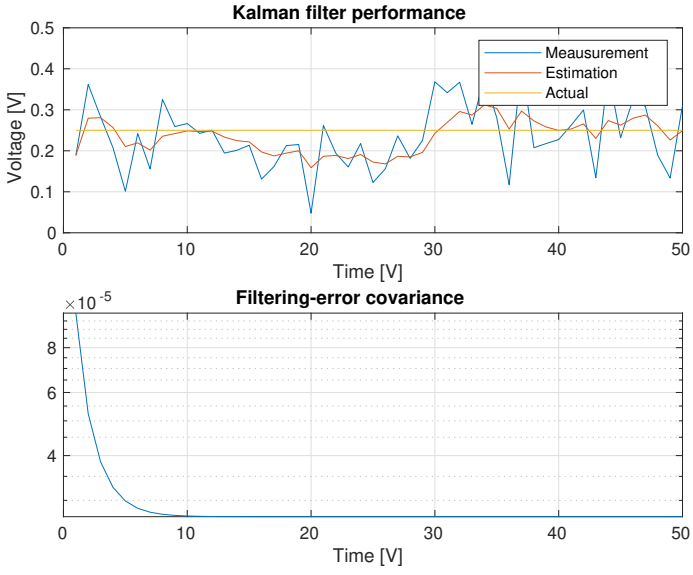

Figure 3: $Q_\nu = 0.01Q_0$

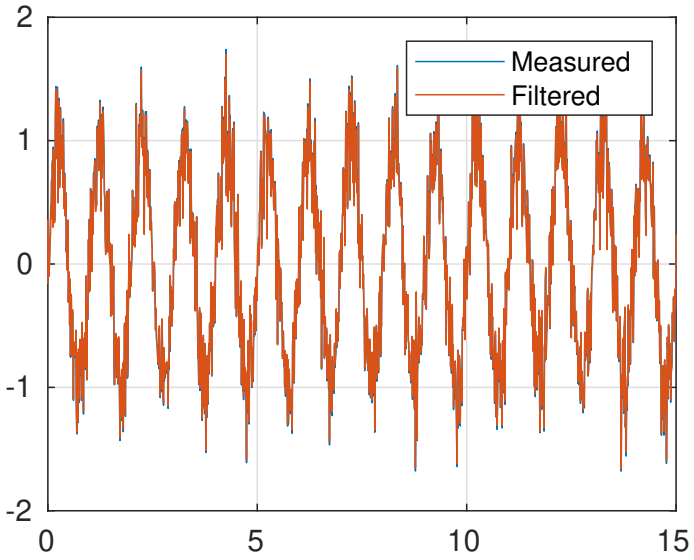

Figure 4: Measured and filtered output in the model of the filtered harmonic oscillator.

5 Appendix A: MATLAB code

```
% Known parameters
y_true = 0.25;
ny_n = 0.1;
Q_w = 1e-5; % Covariance of the dynamic noise
Q_ny = 0.1^2; % Covariance of the measurement noise

% Generate measurement data
rng(2023) % Seed for noise generation
measurement_noise = ny_n * randn(50, 1);
y = y_true * ones(50, 1) + measurement_noise;

% Initialize state estimate and prediction-error
    covariance
x = 0.5;
P = 0.1;

% Initialize filtered series arrays
P_historic = zeros(size(y));
y_filtered = zeros(size(y));

% Loop through measurements
for i = 1:length(y)

    x_pred = x; % No change predicted since constant
        dynamics
    P_pred = P + Q_w; % Update for the prediction-
        error covariance

    G = P_pred / (P_pred + Q_ny); % Calculating Kalman
         gain
    x = x_pred + G * (y(i) - x_pred); % Update based
        on certainty
    P = (1 - G) * P_pred;

    % Store filtered measurement
    P_historic(i) = P;
    y_filtered(i) = x;

end
```

6 Appendix B: MATLAB code

```
measurements = load("harmonic_osc_meas.mat")

A = [0 1; -(2*pi)^2 0];
B = [1 0];

sigma_ny = 0.25; % stdev of the measurement noise


[t_test_clean, y_test_clean] = ode45(@
   harmonic_oscillator, linspace(0,10,1000), [1, 0]);
[t_test_noise, y_test_noise] = ode45(@
   noisy_harmonic_oscillator, linspace(0,10,1000), [1,
    0]);

noise_effect = y_test_noise - y_test_clean
size(noise_effect)

Q_omega = eye(2)%cov(noise_effect);
Q_ny = sigma_ny^2; % because uncorrelated in time

t_meas = measurements.measurements(:,1);
y_meas = measurements.measurements(:,2);

% Define the initial state estimate x_hat_k
x_hat_k = [0; 0];

% Define the initial error covariance matrix P_k
P_k = [1 0; 0 1];

% Initialize variables to store filtered data
y_filtered = zeros(length(t_meas));
P_historic = zeros(2,2,length(t_meas));


x = eye(2);
P = 0.1*eye(2);

% Loop through measurements
for i = 1:length(y_meas)

    x_pred = A*x;
    P_pred = A*P*A' + Q_omega; % Update for the
```

```
        prediction-error covariance

    G = P_pred*B' / (B*P_pred*B' + Q_ny); %
        Calculating Kalman gain
    x = x_pred + G * (y_meas(i) - B*x_pred); % Update
        based on certainty
    P = (eye(2) - G*B) * P_pred;

    % Store filtered measurement
    P_historic(:,:,i) = P;
    y_filtered(:,i) = B*x*B';

end
```