Navigation and Intelligent Vehicles

Lab session 2: evaluating a KF and motion models

In this session, you will learn how to
a) evaluate the consistency of a Kalman Filter
b) implement a Kalman Filter for navigation purposes, using several motion models

We will use a simple one-dimensional motion system that has the position and velocity as a state variable, i.e., $x(k) = [\xi(k) \quad \dot{\xi}(k)]^T$. The state space equations are given by:

$$x(k+1) = Ax(k) + Bu(k) + Gw(k) \tag{1}$$

$$z(k) = Cx(k) + Hv(k) \tag{2}$$

One alternative to model the motion with equation (1) is setting $A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $C = [1 \quad 0]$, $G = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}$ and $H = 1$. With this motion model, the process noise $w(k)$ has an influence on the states evolution, but the input $u(k)$ does not have any influence. The variable T is the sample interval, and will be assumed as unity in the sequel. Later in this document we will describe two motion models.

The scalar process noise $w(k)$ represents the acceleration and is a zero-mean white sequence with variance $E[w^2(k)] = q$. The measurement is corrupted by additive noise $v(k)$ which is a zero-mean white sequence with variance $E[v^2(k)] = r$. We will fix $r = 1$. The two noise sequences are mutually independent.

Kalman filter, enhancing our intuition

As an introductory example, we generate 100 measurement points for each of the following three variances of the process noise: $q = [0, 1, 9]$. Each case produces a different motion. For the first case $q = 0$, there is no change in the acceleration and the velocity is constant. On the other hand, the acceleration with $q = 9$ has a strong impact on the velocity and the position.

We represent the data in a simple way with a state trajectory. Often this gives more insight than plotting a certain state variable against time or sample point. In Figure 1, we observe the true and the estimated trajectories in the velocity-position space for the three cases. Notice that the scales of the axes are different.

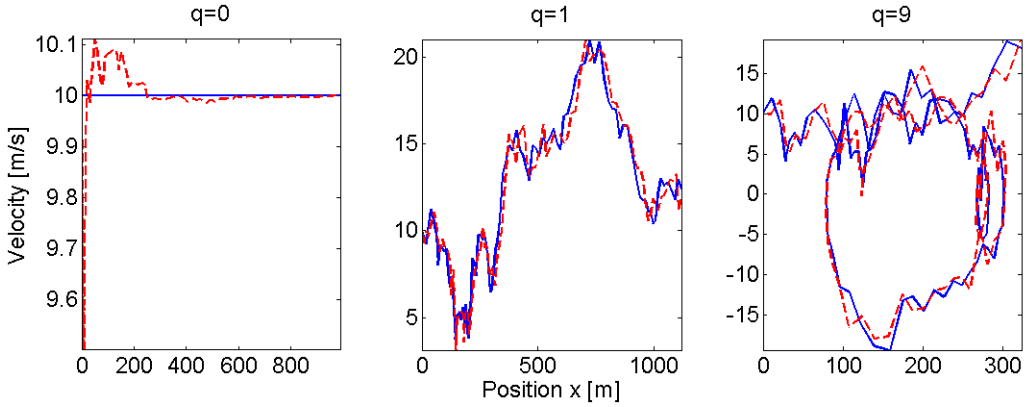


Figure 1: True (solid) and estimated (dashed) state trajectories for process noise variances $q = 0$, 1, and 9.

Kalman filter initialization

To initialize the Kalman filter we use the two-points differentiating method. For this, we use two measurements $[z(1), z(2)]$ to calculate the initial state $\hat{x}_0$ and initial covariance P_0:

$$\hat{x}_0 = \begin{bmatrix} z(1) \\ \frac{z(2)-z(1)}{T} \end{bmatrix} \tag{3}$$

$$P_0 = \begin{bmatrix} R & R/T \\ R/T & 2R/T^2 \end{bmatrix}^{-1} \tag{4}$$

Remark: more than two points are required for the filter initialization if your model has more than two states.

Error covariance interpretation

We have seen in lab session 1 that the error covariance P is a good measure for the variance on the estimated state. For our 1-D motion system, we'll observe the variances of the predicted position P_{11}^- and velocity P_{22}^-, as well as the variances of the estimated position P_{11} and velocity P_{22}.

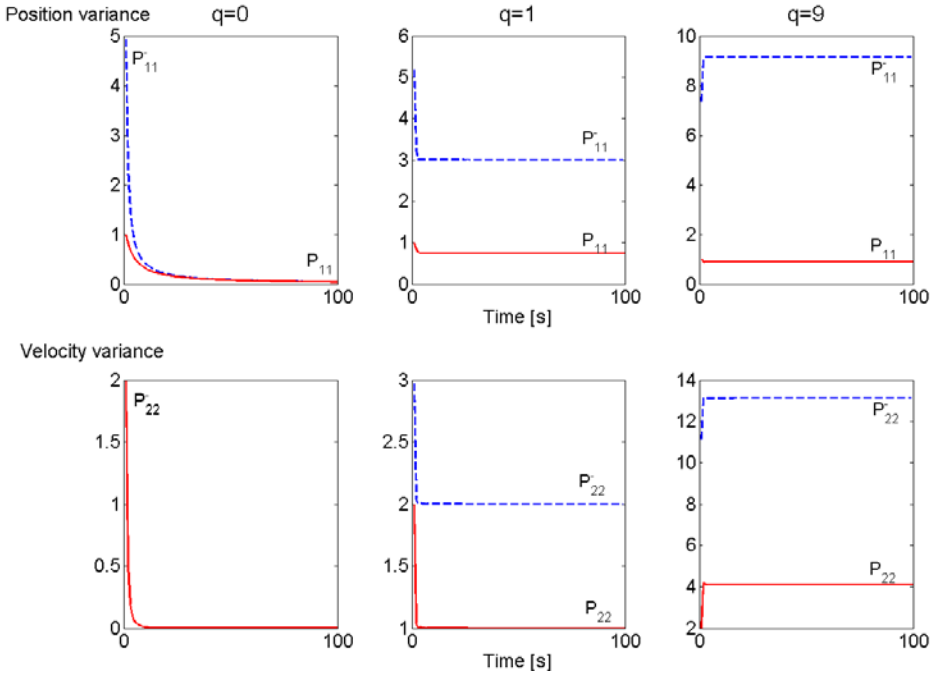


Figure 2: Predicted (dashed) and updated (solid) covariances for $q = 0, 1$ and 9.

For $q = 0$, the estimation covariance matrices converge to zero, as well as the filter gain K. The filter "believes" that it has a perfect state estimation, based on the assumption of a perfect constant velocity motion. Then, the filter shuts itself off. In practice this last assumption usually does not hold, except for short periods of time. For the two other cases, the filter reaches steady state in a few time steps.

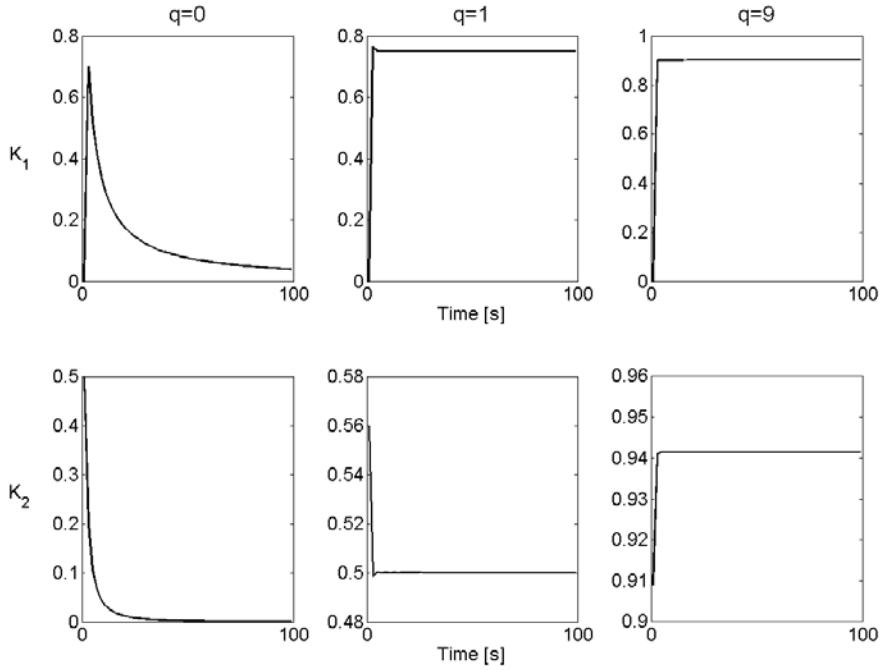


Figure 3: Kalman gain for $q = 0, 1$ and 9.

Kalman gain interpretation

When we examine the steady-state filter gain $\lim_{k\to N} K(k)$ (Figure 3), we see that $K = [0.75 \quad 0.5]^T$ for $q = 1$ and $K = [0.9 \quad 0.94]^T$ for $q = 9$. The higher gains in the second case are caused by the higher process noise level q. This leads to a larger state variance, and thus less accurate state estimates.

Properties of state estimators

In parameter estimation problems, **consistency** of an estimator is defined as **convergence of the estimate to the true value**. This means that on average, we can be sure that the parameter estimation will be arbitrarily close to the true one by collecting enough data.

When estimating or tracking the state of a system, in general no convergence of its estimate occurs, since its value changes over time. What one has, in addition to the "current" estimate of the state, is the associated covariance matrix. This will be used in the filter consistency definition.

A Kalman filter has the following nice properties if the model representation is exact and the initial state and its covariance used in the algorithm are the exact ones:

a) **Unbiasedness** of the estimates, i.e. zero-mean estimation errors:

$$E[x(k) - \hat{x}(k)] = E[\tilde{x}(k)] = 0 \qquad \text{and} \qquad E[x(k) - \hat{x}^{-}(k)] = 0 \tag{5}$$

b) **Covariance matching**, i.e. that the actual mean square error (left hand side) matches the filter-calculated (right hand side) covariances:

$$\text{cov}\big(\tilde{x}(k)\big) = P(k) \qquad \text{and} \qquad \text{cov}\big(x(k) - \hat{x}^{-}(k)\big) = P^{-}(k) \tag{6}$$

c) The **innovations** have zero mean $E[\tilde{z}(k)] = 0$ and have covariance $\text{cov}\big(\tilde{z}(k)\big) = CP^{-}(k)C^{T} + HRH^{T}$.

d) The sequence of **innovations** is white.

A Kalman filter satisfying the above properties (a-d) is called **consistent**. Since the filter gain is based on the filter-calculated error covariances, it follows that consistency is necessary for estimator optimality: wrong covariances yield wrong gain. This is why consistency evaluation is vital for verifying a filter design.

Properties a) and b) can only be checked in simulations, since they assume knowledge of the true state x.

Divergence of a filter means that the state estimation errors are unacceptably large. Divergence can be due to modelling errors or programming errors.

What is an acceptable estimation error will be discussed in the sequel.

In this lab…

We consider the three possible ways of checking the sensitivity to inexact covariance knowledge:

1. single simulation,
2. multiple simulations, and
3. real time tests.

In each situation, two cases will be dealt with:

matched model: the filter uses the exact process model parameters (including the same noise values)

mismatched model: the filter uses wrong process model parameters (different q values, filter uses q_f, but data is generated with q_g).

1. **Single simulation**

A simulation is an offline test of the Kalman filter where the true state values are known and thus can be compared with. For this, the following two indicators are very often used:

NEES – the normalised (state) estimation error squared

This is an indicator where the ISE and P are combined into one indicator:

$$\text{NESS}(k) = \tilde{x}(k)^T P(k)^{-1} \tilde{x}(k) \qquad \text{with} \qquad \tilde{x}(k) = x(k) - \hat{x}(k) \tag{7}$$

If the error $\tilde{x}$ is Gaussian distributed (basic assumption of the Kalman filter), the NEES is chi-square distributed, with the degrees of freedom equal to the number of states.

This indicator is used as a metric to check whether the state estimation is unbiased and whether the state errors are normally distributed with covariance P.

NIS – the normalised innovation squared

This indicator doesn't give as much information as the NEES, but its main advantage is that it can be used in online situations where we don't know the true state value: only the output measurement is used.

$$\text{NIS}(k) = \tilde{z}(k)^T S(k)^{-1} \tilde{z}(k) \quad \text{with} \quad \tilde{z}(k) = z(k) - C\hat{x}^-(k) \quad \text{and} \quad S(k) = CP(k)^- C^T + HRH^T \tag{8}$$

Here as well, a Gaussian distribution of $\tilde{z}$ will lead to a chi-square distribution of the NIS. The degree of freedom is the same as the dimension of the measurement variable.

Probability region

For both the NEES and NIS, the obtained values should not exceed a certain threshold, which can be calculated based on their chi-square distribution. It is common to take the 95% probability region.

The Figure 4 shows the behaviour of NEES for the process noise variances $q = 0, 1,$ and 9 for filters that are perfectly matched. When the filters are perfectly matched, only few of 100 points are outside the 95% probability region, that is perfectly acceptable. One could remark that it is useless to compute the probability region for $q = 0$ since the distribution is in fact not chi-square. Choosing $q = 0$ is anyhow undesirable because the covariance estimates will converge to zero.
The case is different for the NEES of a mismatched filter. Nearly half of the points are lying outside of the 95% probability region, which is clearly an unacceptable situation.
The one-sided probability region was determined as follows: we have a $n_x = 2$ degrees of freedom chi-square random variable, so the 5% tail point is: $\chi_2^2(0.95) = 5.99 \approx 6$.

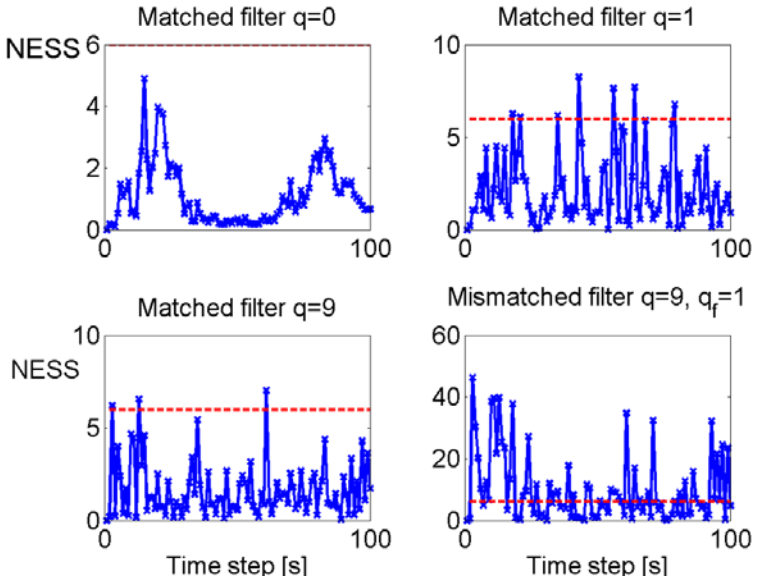


Figure 4: Single-run NEES for matched filters with $q = 0$, 1, and 9, and a single-run for a mismatched filter with $q_g = 9$ and $q_f = 1$

2. Multiple simulations, Monte Carlo principle

The performance of an estimation (or control) algorithm is usually the expected value $\bar{\varepsilon} = E[\varepsilon]$ of a cost function ε. If ε is, for instance, the squared error in the estimation of a certain variable, then $\bar{\varepsilon}$ is the corresponding mean squared error. In this lab, ε can be the NEES or NIS indicators.

In many situations, the performance of an algorithm of interest cannot be evaluated analytically. In such a case, Monte Carlo simulations (runs) are made to obtain an estimate of $\bar{\varepsilon}$ from a sample average of independent realisations ε_i for $i = 1, \dots, N_{MC}$, of the cost function ε. The larger the number of such runs, the smaller is the variability (error) of the resulting estimate.

$$\bar{\varepsilon} = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \varepsilon_i(k) \tag{9}$$

The tests in the previous section were based on data from a single run and provided valuable information. Applying the same test to N_{MC} runs with **independent random variables** we can draw **more generalised conclusions**.

NEES and NIS

We run N_{MC} simulations to obtain N independent samples $\varepsilon_i(k)$ for $i = 1, \dots, N_{MC}$, where the random variable is $\varepsilon_i(k) = \text{NEES}(k)$ or $\varepsilon_i(k) = \text{NIS}(k)$. Then we compute the ($N$-run) average $\bar{\varepsilon}(k)$.

Under the hypothesis that the filter is consistent, it can be shown that the averaged NEES and NIS have a chi-square distribution with Nn_x and Nn_z degrees of freedom, respectively, where n_x is the dimension of the state variable x and n_z is the dimension of the measurement variable z.

SAC – the sample autocorrelation

It is also possible to test the whiteness of the innovations (see property d). Let's define the SAC as:

$$\bar{\rho}(k,j) = \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \tilde{z}_i(k)^T \tilde{z}_i(j) \cdot \left[\frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \tilde{z}_i(k)^T \tilde{z}_i(k) \sum_{i=1}^{N_{MC}} \tilde{z}_i(j)^T \tilde{z}_i(j) \right]^{-\frac{1}{2}} \tag{10}$$

The variables k and j are for choosing two measurement points in the realisation. For N_{MC} large enough, a normal approximation of the density of $\bar{\rho}$ for $k \neq j$ is convenient and reasonable in view of the central limit theorem. If the innovations are zero-mean and white, then the mean $\bar{\rho}$ is zero and its variance is $1/N_{MC}$.

Confidence interval calculation

The 95% intervals are marked with the dashed red lines on the Figures 5 and 6. They are calculated as:

For the NEES: the two-sided 95% region for a chi-square random variable of 100 degrees of freedom (2 dimensions in the state times 50 runs) is $[\chi^2_{100}(0.025) \quad \chi^2_{100}(0.975)] = [74.2 \quad 129.6]$. Dividing these values by $N_{MC} = 50$ leaves us with the 95% probability region for the average normalized state estimation error: $[1.5 \quad 2.6]$.

For the NIS: the two-sided 95% region for a chi-square random variable of 50 degrees of freedom (1 dimension times 50 runs) is $[\chi^2_{50}(0.025) \quad \chi^2_{50}(0.975)] = [32.3 \quad 71.4]$. Dividing by $N_{MC} = 50$ gives us $[0.65 \quad 1.43]$.

For the SAC: we have plotted the innovations one step apart ($k - j = 1$). The 95% region $[-1.96\sigma \quad 01.96\sigma]$ is the interval $[-0.277 \quad 0.277]$.

We see that for the correct filter in Figure 5, an acceptable low amount of points (out of 100) fall outside of the 95% region, which is clearly not the case for the mismatched filter in Figure 6.

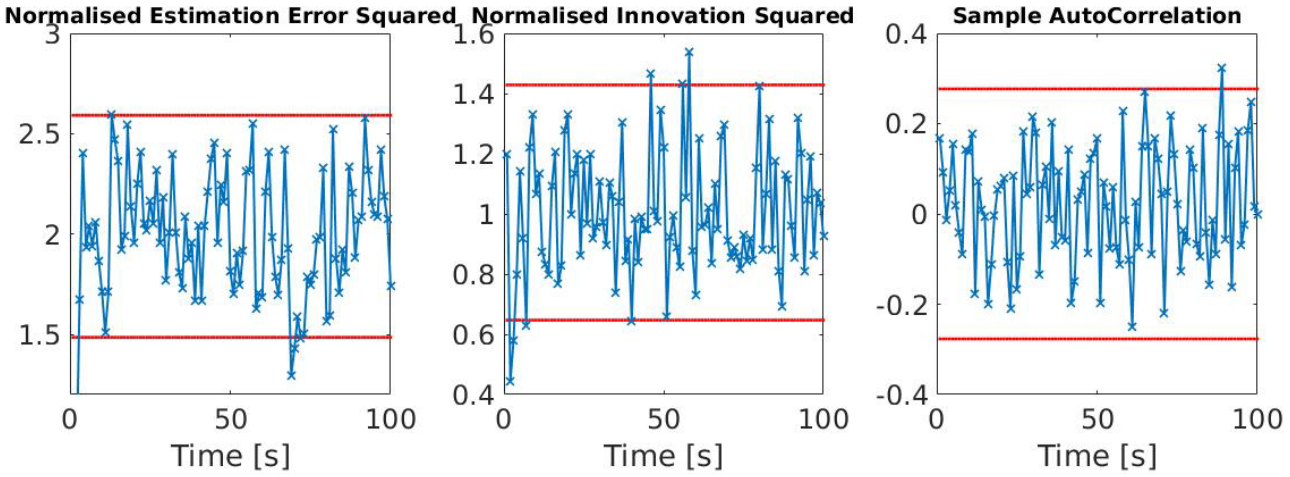


Figure 5: Results from 50 Monte Carlo runs with $q_g = q_f = 1$.

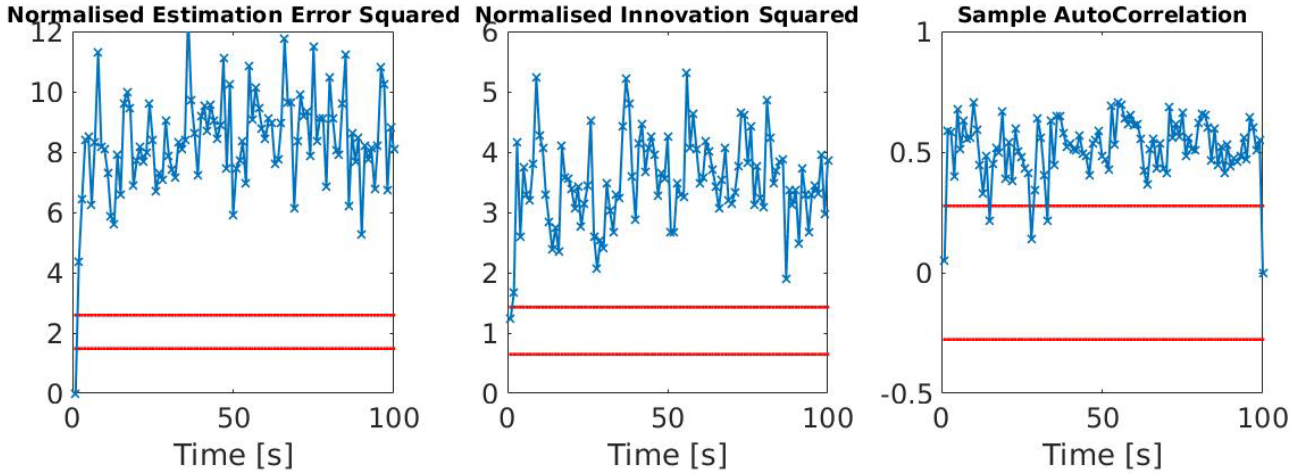


Figure 6: Results from 50 Monte Carlo runs with the mismatched filter $q_g = 9$ and $q_f = 1$.

Filter tuning

The process noise is used in practice to model disturbances, e.g. unknown inputs like target manoeuvres in tracking. The procedure to match the process noise variance to suitably model such disturbances is called filter tuning. Since such inputs are not real noise in the probabilistic sense, it is said that the filter uses pseudo-noise or artificial noise.

The procedure for tuning is to make the filter consistent. While this is easier said than done, in practice one has to strive to make the filter as close to being consistent as feasible, while achieving small RMS estimation errors, at the same time.

For example, if the NEES is too high, this might be caused by a bias in the state estimation error. A separate bias test can then be performed, by taking each component of the state error and testing to see if its mean can be accepted as zero. By dividing it by its standard deviation (rendering it – under ideal conditions – $N(0,1)$), this can be made easier.

3. Real time tests

A real time test is a test that is performed online during a single run, thus while the filter is in operation. We have seen already that for optimal results, the above tests assume that N independent runs have been made. These tests can be used on a single run ($N=1$), but have a higher variability in this case. The question now is whether one can achieve a low variability of the test statistic based on a single run; that is having a **real time consistency test**.

Since the true state value is unknown, only the innovation properties can be checked (see Kalman filter properties c and d). To do so, we **replace the ensemble averages by time averages** based on the stationarity and ergodicity of the innovation sequence (which can be assumed).

TA-NIS (time-average normalised innovation squared)

We can define a so-called TA-NIS statistic as:

$$\bar{\varepsilon}_{\tilde{z}} = \frac{1}{K}\sum_{k=1}^{K} \tilde{z}(k)^T S(k)^{-1} \tilde{z}(k) \tag{11}$$

If the innovations have zero mean and covariance $S(k)$, then $K\bar{\varepsilon}_{\tilde{z}}$ has a chi-square distribution with Kn_z degrees of freedom.

TA-AC (time-average autocorrelation)

The whiteness test statistic for innovations ℓ steps apart from a single run can be written as the TA-AC:

$$\bar{\rho}(l) = \sum_{k=1}^{K} \tilde{z}(k)^T \tilde{z}(k+l) \left[\sum_{k=1}^{K} \tilde{z}(k)^T \tilde{z}(k) \sum_{k=1}^{K} \tilde{z}(k+l)^T \tilde{z}(k+l)\right]^{-1/2} \tag{12}$$

This statistic is, for large enough K, in view of the central limit theorem, normally distributed. Furthermore, its variance can be shown to be $1/K$.

The probability region for acceptance of the "consistent filter" hypothesis is set up as before.

We calculated the TA-AC and TA-NIS both for a matched and mismatched case, see Table 1.

Table 1: Time-average test statistic values, obtained from 100 samples in time

	TA-AC	TA-NIS
Correct filter	$\bar{\rho}(1) = 0.152$	$\bar{\varepsilon}_{\tilde{z}} = 0.936$
Mismatched case	$\bar{\rho}(1) = 0.509$	$\bar{\varepsilon}_{\tilde{z}} = 2.66$

Under the assumption that the filter is correct, for the TA-AC the error is normally distributed with mean zero and variance 0.01, and its 95% probability region is $[-0.196, 0.196]$. It is clear that the correct filter falls in this region (as expected), where the result for the mismatched case is clearly unacceptable.

For the TA-NIS (attention: ideal value is 1), the 95% confidence region is, based on the 100 degrees of freedom chi-square distribution, the interval $[0.74, 1.3]$. We can again draw the same conclusions.

Motion models

Over the years, many motion models have been developed. We will present two:

1. the constant velocity, and
2. the constant acceleration model.

For both, there exists a variant that is derived from the continuous time (integrated) and a discrete time (piecewise constant) model.

1. Constant velocity model

The constant velocity model assumes that the vehicle travels at an approximately constant velocity. Changes in the velocity are modelled by a zero-mean process noise representing acceleration.

White noise acceleration model

For tracking in 1-D, the process equation is given by

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{w}_k, \qquad \mathbf{w}_k \sim N\left(\begin{bmatrix}0 & 0\end{bmatrix}^T, \mathbf{Q}\right) \tag{13}$$

where $\mathbf{x}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$, $\mathbf{A} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$, and $\mathbf{Q} = \sigma_a^2 \begin{bmatrix} T^3/3 & T^2/2 \\ T^2/2 & T \end{bmatrix}$, T is the sampling period and σ_a^2 is the acceleration variance.

Piecewise constant, white noise acceleration model

If w_k is the constant acceleration during the *k*-th sampling period (of length T), the increment in the velocity during this period is $w_k T$. Moreover, the effect of this acceleration on the position is $w_k T^2/2$. The state equation for the piecewise constant white acceleration mode is then

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}\mathbf{x}_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} w_k, \qquad w_k \sim N(0, \sigma_a^2) \tag{14}$$

2. Constant acceleration model

The constant acceleration model assumes that the tracked object has a quasi-constant acceleration, with the small acceleration variations being modelled as a zero-mean noise-process, $\dddot{x} = \eta(t)$.

Constant acceleration model a.k.a. Wiener process acceleration model

The system evolution is identical to (13), but the state vector is three-dimensional and the state transition matrix and the process noise covariance matrix have the following expressions:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{Q} = \sigma_a^2 \begin{bmatrix} T^5/20 & T^4/8 & T^3/6 \\ T^4/8 & T^3/3 & T^2/2 \\ T^3/6 & T^2/2 & T \end{bmatrix}. \tag{15}$$

Piecewise constant Wiener process acceleration model

In this model, the process noise w_k is the acceleration increment during the *k*-th sampling period and it is assumed to be a zero-mean white sequence – the acceleration is a discrete time Wiener process. The state equation is:

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} T^2/2 \\ T \\ 1 \end{bmatrix} w(k), \qquad w(k) \sim N(0, \sigma_a^2) \tag{16}$$

Model analysis and report tasks

The knowledge gained from the theory will be extended by evaluating a KF and motion models presented above. We put a focus on different situations: single simulation, sequence of simulations and real time tests.

Firstly, in the single simulation you can use the concept/code that is made in the Lab 1 as a base, modifying the code and adapting it towards a two state simulation**.** Next, you extend the code to a sequence of simulations (Monte Carlo principle). To start doing this lab, generate your own data by running the script *generate_data.m* with a certain value of *r*.

For determining the confidence intervals you can either use a table with the chi-square values, or the Matlab built-in function *chi2inv(P, Dof),* where *P* is the confidence interval and *DoF* the number of degrees of freedom.

Summation for the tasks required in the lab 2 is as follows:

1. **Constant velocity model (equations 13 and 14)**
    a. Single run simulation
    b. Multiple simulations (Monte Carlo)
2. **Constant acceleration model (equations 15 and 16)**
    a. Single run simulation
    b. Multiple simulations (Monte Carlo)

- For both models make sure you use 4 different values for q , ranging from 0 to 10, such as 0,0.1,1,10. *R* can be assumed to be known.

- Use NIS, NEES, SAC indicators and the real time tests TA-NIS, TA-AC for analysing your data and figures.

Make a short report (maximum six pages) in which you summarize (text) and illustrate (graphs) the performance of both models.

- In case you generated your own data, state clearly which simulation settings you used!
- State clearly which initialization (x_0, $\hat{x}_0$, P_0) and which KF parameters are used!
- Provide and check the model structure, define correctly and provide the matrices.

References

Yaakov Bar-Shalom, Xiao-Rong Li, "Estimation and Tracking, Principles, Techniques, and Software", Artech House Publishing 1993, ISBN 0-89006-643-4

Appendix I: Derivation of white noise acceleration model

In the following, we show how (13) is obtained from a continuous-time model. Consider a continuous-time model where the acceleration is a zero-mean noise process, $\ddot{x} = a(t) \sim N(0, \sigma_a^2)$. With the state vector comprising the position and the velocity at the moment t, the system evolution equation is as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \eta(t) \qquad \eta(t) \sim N(0, \Upsilon)$$

where the state vector is given by $\mathbf{x}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$, the state update matrix by $\mathbf{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, the process noise term by $\eta(t) = \begin{bmatrix} 0 \\ a(t) \end{bmatrix}$, and the noise covariance by $\Upsilon = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}$.

A discrete model can be derived by using a sampling interval $T_k = t_{k+1} - t_k$. For a time-invariant system, the transition and noise matrices can be found from the continuous model as follows:

$$\mathbf{A}_k = e^{\mathbf{F}(t_{k+1} - t_k)}$$

$$\mathbf{w}_k = \int_{t_k}^{t_{k+1}} e^{\mathbf{F}(t_{k+1} - \tau)} \eta(\tau) d\tau$$

For the particular form of $\mathbf{F}$ in our model, using a Taylor series development and keeping only the first order terms yields

$$\mathbf{A}_k = e^{\mathbf{F}T_k} = \mathbf{I} + \mathbf{F}T_k + \mathbf{F}^2 \frac{T_k^2}{2!} + \mathbf{F}^3 \frac{T_k^3}{3!} + \cdots \approx \begin{bmatrix} 1 & T_k \\ 0 & 1 \end{bmatrix},$$

where $\mathbf{A}_k$ is the state transition matrix describing the system evolution during $[t_k, t_{k+1}]$.

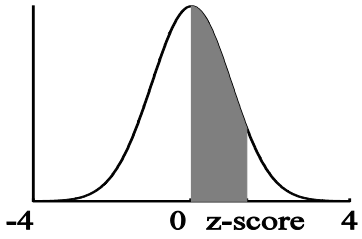
The discrete-time process noise covariance is obtained as

$$\mathbf{Q}_k = \int_{t_k}^{t_{k+1}} e^{\mathbf{F}(t_{k+1} - \tau)} \Upsilon e^{\mathbf{F}^T (t_{k+1} - \tau)} d\tau \approx \int_{t_k}^{t_{k+1}} \begin{bmatrix} 1 & t_{k+1} - \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & \sigma_a^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_{k+1} - \tau & 1 \end{bmatrix} d\tau = \sigma_a^2 \begin{bmatrix} T_k^3/3 & T_k^2/2 \\ T_k^2/2 & T_k \end{bmatrix}.$$

Appendix II: Probability lookup table: Standard normal distribution

The standard normal distribution $N(0,1)$ has the following properties:

Mean is zero
Variance is one (thus the standard deviation too)
Data values are represented by z



The probability function is given by: $p(z) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{z^2}{2}}$

To convert from a normal distributed $N(\mu,\sigma)$ variable to a standard normal distributed variable z, you can use the following formula:

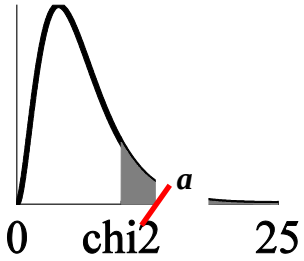
$$z = \frac{(x-\mu)}{\sigma}$$

Table 2: the values in the table are the areas between zero and the z-score. That is, P(0<Z<z-score)

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0	0.004	0.008	0.012	0.016	0.0199	0.0239	0.0279	0.0319	0.0359
0.1	0.0398	0.0438	0.0478	0.0517	0.0557	0.0596	0.0636	0.0675	0.0714	0.0753
0.2	0.0793	0.0832	0.0871	0.091	0.0948	0.0987	0.1026	0.1064	0.1103	0.1141
0.3	0.1179	0.1217	0.1255	0.1293	0.1331	0.1368	0.1406	0.1443	0.148	0.1517
0.4	0.1554	0.1591	0.1628	0.1664	0.17	0.1736	0.1772	0.1808	0.1844	0.1879
0.5	0.1915	0.195	0.1985	0.2019	0.2054	0.2088	0.2123	0.2157	0.219	0.2224
0.6	0.2257	0.2291	0.2324	0.2357	0.2389	0.2422	0.2454	0.2486	0.2517	0.2549
0.7	0.258	0.2611	0.2642	0.2673	0.2704	0.2734	0.2764	0.2794	0.2823	0.2852
0.8	0.2881	0.291	0.2939	0.2967	0.2995	0.3023	0.3051	0.3078	0.3106	0.3133
0.9	0.3159	0.3186	0.3212	0.3238	0.3264	0.3289	0.3315	0.334	0.3365	0.3389
1	0.3413	0.3438	0.3461	0.3485	0.3508	0.3531	0.3554	0.3577	0.3599	0.3621
1.1	0.3643	0.3665	0.3686	0.3708	0.3729	0.3749	0.377	0.379	0.381	0.383
1.2	0.3849	0.3869	0.3888	0.3907	0.3925	0.3944	0.3962	0.398	0.3997	0.4015
1.3	0.4032	0.4049	0.4066	0.4082	0.4099	0.4115	0.4131	0.4147	0.4162	0.4177
1.4	0.4192	0.4207	0.4222	0.4236	0.4251	0.4265	0.4279	0.4292	0.4306	0.4319
1.5	0.4332	0.4345	0.4357	0.437	0.4382	0.4394	0.4406	0.4418	0.4429	0.4441
1.6	0.4452	0.4463	0.4474	0.4484	0.4495	0.4505	0.4515	0.4525	0.4535	0.4545
1.7	0.4554	0.4564	0.4573	0.4582	0.4591	0.4599	0.4608	0.4616	0.4625	0.4633
1.8	0.4641	0.4649	0.4656	0.4664	0.4671	0.4678	0.4686	0.4693	0.4699	0.4706
1.9	0.4713	0.4719	0.4726	0.4732	0.4738	0.4744	0.475	0.4756	0.4761	0.4767
2	0.4772	0.4778	0.4783	0.4788	0.4793	0.4798	0.4803	0.4808	0.4812	0.4817
2.1	0.4821	0.4826	0.483	0.4834	0.4838	0.4842	0.4846	0.485	0.4854	0.4857
2.2	0.4861	0.4864	0.4868	0.4871	0.4875	0.4878	0.4881	0.4884	0.4887	0.489
2.3	0.4893	0.4896	0.4898	0.4901	0.4904	0.4906	0.4909	0.4911	0.4913	0.4916
2.4	0.4918	0.492	0.4922	0.4925	0.4927	0.4929	0.4931	0.4932	0.4934	0.4936
2.5	0.4938	0.494	0.4941	0.4943	0.4945	0.4946	0.4948	0.4949	0.4951	0.4952
2.6	0.4953	0.4955	0.4956	0.4957	0.4959	0.496	0.4961	0.4962	0.4963	0.4964
2.7	0.4965	0.4966	0.4967	0.4968	0.4969	0.497	0.4971	0.4972	0.4973	0.4974
2.8	0.4974	0.4975	0.4976	0.4977	0.4977	0.4978	0.4979	0.4979	0.498	0.4981
2.9	0.4981	0.4982	0.4982	0.4983	0.4984	0.4984	0.4985	0.4985	0.4986	0.4986
3	0.4987	0.4987	0.4987	0.4988	0.4988	0.4989	0.4989	0.4989	0.499	0.499

Appendix III: Probability lookup table: Chi-square distribution

The χ_a^2 distribution is not symmetric; so looking up left-tail values is different from looking up right-tail values.



Area to the right – just use the area given.
Areas to the left - subtract the given area from one and look this area up in the table.
Area in both tails - divide the area by two. Look up this area for the right critical value and one minus this area for the left critical value.

When a degree of freedom (dof) is not listed in the table you:

interpolate, this is probably the more accurate way. (Linear) interpolation involves figuring out how far the given dof is between the two dof's in the table and correcting the value proportionally; or
use the critical value, which is less likely to cause you to reject an error. For a right tail test, this is the critical value further to the right (larger). For a left tail test, it is the value further to the left (smaller). For a two-tail test, it's the value further to the left and the value further to the right.

Table 3: The areas given across the top are the areas to the right of the critical value χ_a^2

dof	$\chi^2_{0.995}$	$\chi^2_{0.99}$	$\chi^2_{0.975}$	$\chi^2_{0.95}$	$\chi^2_{0.9}$	$\chi^2_{0.1}$	$\chi^2_{0.05}$	$\chi^2_{0.025}$	$\chi^2_{0.01}$	$\chi^2_{0.005}$
1	---	---	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.01	0.02	0.051	0.103	0.211	4.605	5.991	7.378	9.21	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.86
5	0.412	0.554	0.831	1.145	1.61	9.236	11.07	12.833	15.086	16.75
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	0.989	1.239	1.69	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.646	2.18	2.733	3.49	13.362	15.507	17.535	20.09	21.955
9	1.735	2.088	2.7	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.94	4.865	15.987	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	5.578	17.275	19.675	21.92	24.725	26.757
12	3.074	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217	28.3
13	3.565	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688	29.819
14	4.075	4.66	5.629	6.571	7.79	21.064	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32	34.267
17	5.697	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.39	10.865	25.989	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191	38.582
20	7.434	8.26	9.591	10.851	12.443	28.412	31.41	34.17	37.566	39.997
21	8.034	8.897	10.283	11.591	13.24	29.615	32.671	35.479	38.932	41.401
22	8.643	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289	42.796
23	9.26	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638	44.181
24	9.886	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.98	45.559
25	10.52	11.524	13.12	14.611	16.473	34.382	37.652	40.646	44.314	46.928
26	11.16	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642	48.29
27	11.808	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963	49.645
28	12.461	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278	50.993
29	13.121	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588	52.336
30	13.787	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892	53.672
40	20.707	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691	66.766
50	27.991	29.707	32.357	34.764	37.689	63.167	67.505	71.42	76.154	79.49
60	35.534	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379	91.952
70	43.275	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425	104.215
80	51.172	53.54	57.153	60.391	64.278	96.578	101.879	106.629	112.329	116.321
90	59.196	61.754	65.647	69.126	73.291	107.565	113.145	118.136	124.116	128.299
100	67.328	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807	140.169