Course on:

Optimal estimation in dynamic systems

Exercises

date: 20 October 2020

F. van der Heijden

mailto:F.vanderHeijden@utwente.nl

Carré 3611

This document contains eight exercises that accompany the course on optimal estimation. These exercises run parallel with the lectures. They address the following topics:

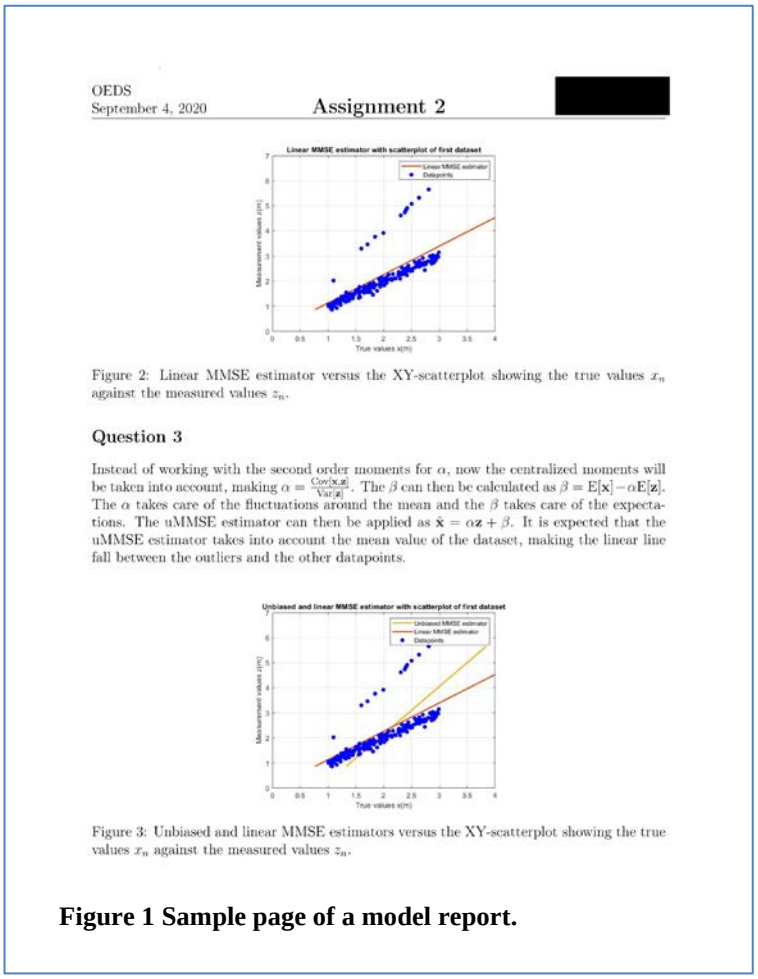
- Estimation of a static, single parameter: exercise 1 and 2
- Estimation of a static parameter vector: exercise 3
- Prediction in a linear dynamic system: exercise 4
- Estimation in a time-invariant, linear Gaussian system: exercise 5
- Estimation in a near-linear, near-Gaussian system: exercise 6 and 7
- Particle filtering exercise 7
- EKF-SLAM exercise 8
- Smoothing exercise 8

Reports

Each exercise is to be concluded with a pdf report:

- For each question in an exercise the report contains:
  - The accomplishment of the question, e.g. a table or figure.
  - An interpretation of the results. Use your academic competence: what did you expect to see, and why? What do you observe? Is that according to your expectation? If not, explain. Are there any other unexpected or otherwise remarkable phenomena which you observe?
- Use an equation editor for all mathematical symbols.
- Graphs should be included with axis labels, ticks, and if necessary, a legend.
- Add the MATLAB code at the end as an appendix integrated in the document.

To provide you with an impression, how the report should look like, a sample page of a model report is shown in Figure 1.



OEDS
September 4, 2020 **Assignment 2** [redacted]

Figure 2: Linear MMSE estimator versus the XY-scatterplot showing the true values x_n against the measured values z_n.

Question 3

Instead of working with the second order moments for α, now the centralized moments will be taken into account, making $\alpha = \frac{\text{Cov}[\mathbf{x},\mathbf{z}]}{\text{Var}[\mathbf{z}]}$. The β can then be calculated as $\beta = \text{E}[\mathbf{x}] - \alpha\text{E}[\mathbf{z}]$. The α takes care of the fluctuations around the mean and the β takes care of the expectations. The uMMSE estimator can then be applied as $\hat{\mathbf{x}} = \alpha\mathbf{z} + \beta$. It is expected that the uMMSE estimator takes into account the mean value of the dataset, making the linear line fall between the outliers and the other datapoints.

Figure 3: Unbiased and linear MMSE estimators versus the XY-scatterplot showing the true values x_n against the measured values z_n.

Figure 1 Sample page of a model report.

Submission

- Submission types: only pdf's and mp4 files.
- You can submit each report separately, or you can bundle all reports into a single pdf.
- In exercise 8, movies are asked. This should be mp4 movies.

Assessment

The course is finalized with an oral exam. Students are graded in a discussion where their knowledge and understanding of the concepts, their ability to apply and to integrate them, and the ability to communicate about these concepts will be tested. This will be done along the reports of the exercises. Therefore, these reports are the guideline of the discussion. The reports themselves are not graded separately, and not graded explicitly in advance. The assessment plan is published on Canvas. See the section Syllabus at the Canvas site.

Exercise 1: Fundamentals of parameter estimation - part I

MAP estimation, MMSE estimation; MMAE estimation; ML estimation

The case that is considered in this exercise is the estimation of the depth of the water below a ship. See Figure 2.

The measurement system: an ultrasonic depth gauge

An ultrasonic depth gauge is a sensor system consisting of an ultrasonic transmitter and receiver, that are mounted under a boat. During the measurement, a tone burst is transmitted in downward direction. The waveform reflects at the bottom of the sea. The echo is picked up by the receiver. The elapse of time between sending the tone burst and receiving the echo is the so-called ToF, the time of flight. The ToF is proportional to the depth x. If c is the speed of sound in water, then $ToF = 2x/c$. The ToF is determined, for instance, simply by comparing the observed echo against a threshold. The moment at which the observed signal exceeds the threshold defines the time of arrival. The measurement is obtained by:

$$z = \frac{c}{2} ToF \tag{1}$$

The measurement may be disturbed by the following phenomena:

- Secondary echoes
  The echo may reflect at the bottom of the boat causing a second echo that arrives at $t = 4x/c$. The second echo may cause a third echo, and so on. See Figure 2. In contrast with the second echo, the third echo and any succeeding echoes are weak and negligible. However, sometimes, the second echo overrules the first echo. In that case, the time-of-flight erroneously becomes $ToF = 4x/c$.
- Electronic noise.
  The measurement is contaminated by Gaussian noise with standard deviation σ. Suppose that x is the real depth, and z is the result of our measurement. Then, we adopt the following gaussian mixture model for the conditional probability density of z:

$$p(z \mid x) = P_0 \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(z-x)^2}{2\sigma^2}\right) + P_1 \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(z-2x)^2}{2\sigma^2}\right) \tag{2}$$

  P_1 is the probability that the first echo is missed and replaced by the second echo. P_0 is the probability that the first echo is correctly detected. Clearly: $P_0 + P_1 = 1$.

The prior knowledge

Based on the information from a nautical map, the shipper has some prior knowledge about the depth. The map indicates an interval of possible depths. This interval is considered to be softly bounded. Such prior knowledge can be modelled with a so-called *generalized normal distribution*:

$$p(x) = \frac{\beta}{2\alpha\,\Gamma(1/\beta)} \exp\left(-\left(\frac{|x-\mu|}{\alpha}\right)^{\beta}\right) \tag{3}$$

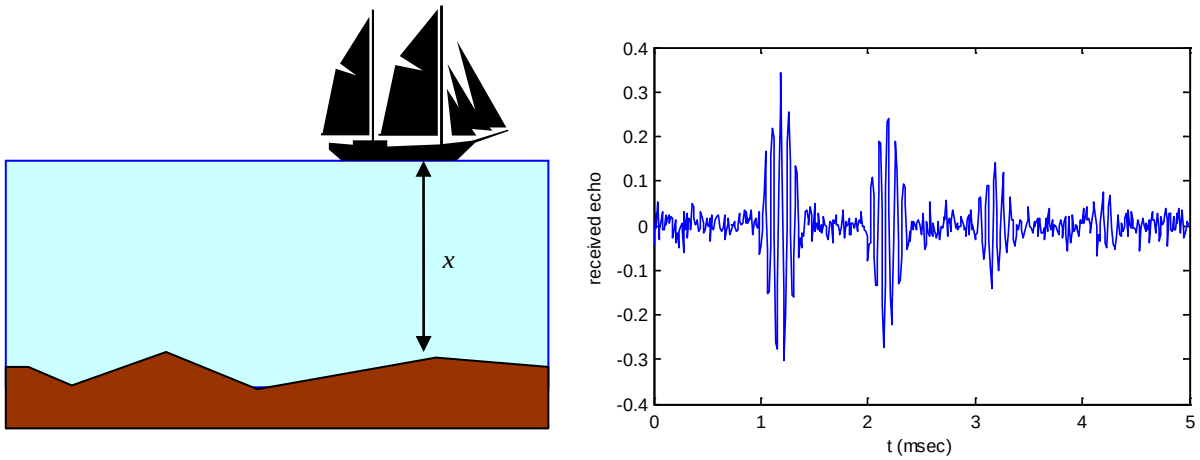


Figure 2 Measuring depth using an ultrasonic depth gauge

μ is the expectation, α is a measure of the width of the distribution: if the depth interval is $[x_{\min}, x_{\max}]$, then $\mu = \frac{1}{2}(x_{\min} + x_{\max})$, and $\alpha = \frac{1}{2}(x_{\max} - x_{\min})$. The steepness of the bounds is defined by β. The distribution becomes Gaussian if $\beta = 2$. The distribution is uniform, i.e. hard bounded if $\beta \to \infty$. $\Gamma()$ is the gamma function.

The case:
For the current case, the following parameters are given:

sensor systems:	$P_0 = 0.95$	prior knowledge:	$x_{\min} = 1$ m
	$\sigma = 0.1$ m		$x_{\max} = 3$ m
			$\beta = 20$

1. Create an m-file `px.m` containing a function `px(x)` that returns the prior pdf* $p(x)$. Create also an m-file `pz_x.m` that defines a function `pz_x(z,x)` that returns the likelihood function $p(z \mid x)$. Use these functions to create a graph of $p(x)$ and graphs of $p(z \mid x)$ against z for $x = 1.5$ m, and for $x = 2$ m. What do these pdfs model?
2. Create also m-files `pz.m` and `px_z.m` that implements the pdfs $p(z)$ and $p(x \mid z)$, respectively. Use these functions to create a graph of $p(z)$, and also graphs of $p(x \mid z)$ against x for $z = 3.1$ m, and for $z = 4$ m. What do these graphs model?
3. Create m-files that calculate:
   - The MMSE estimator $\hat{x}_{MMSE}(z)$ for $z = 3.1$ m and for $z = 4$ m.
   - The MAP estimator $\hat{x}_{MAP}(z)$ for $z = 3.1$ m and for $z = 4$ m.
   - The MMAE estimator $\hat{x}_{MMAE}(z)$ for $z = 3.1$ m and for $z = 4$ m.
   - The ML estimator $\hat{x}_{ML}(z)$ for $z = 3.1$ m, and for $z = 4$ m.

   Can you explain the results, especial the ones for $z = 3.1$ m ?
4. Calculate for each case in 3 the conditional risk. Do that for any of the following cost functions:
   - Quadratic cost function
   - Absolute cost function
   - Uniform cost function with $\Delta = 0.05$ m. The definition of Δ is $C_{uni}(x \mid \hat{x}) = 1$ if $|x - \hat{x}| > \Delta$

   The risks that are calculated may have a physical unit. Don't forget to add them.
5. Compare and explain the results of question 4..

Hints:

You may want to use the following MATLAB functions:
- `gamma` for the gamma function.
- `normpdf` for the normal distribution (requires the Statistics and Machine Learning Toolbox)

The needed integrals can be implemented with simple numerical integrations, e.g. the Euler method as is done in listing 3.1 in the textbook; page 53, 54. However, the listing contains typos:
- Page 54: add 3 dots, `'...'`, at the end of line 3 to continue the statement with line 4.
- Page 54, line 4 and 5: replace `'./.'` with `'./'`
- Page 54, function **`pz`**:

```
global xrange;                          % xrange: array of samples of x
dx = xrange(2)-xrange(1);               % dx is sample period
ret = sum(px(xrange).*pz_x(z,xrange))*dx; % numerical integration
```

* pdf: probability density function

Exercise 2: Fundamentals of parameter estimation - part II

linear MMSE and unbiased linear MMSE

We reconsider the ultrasonic depth gauge discussed in exercise 1. Figure 3 shows two data sets that are obtained from the probabilistic model. Each data set contains 200 realizations (x_n, z_n). The realizations are a-select and independent. The goal of this exercise is:

- To design a linMMSE estimator and an unbiased linMMSE estimator using the first data set.
- To evaluate these estimators with respect to bias and variance using the second data set.

1. Load the first data set (Matlab: `load depthgauge_data_set1.mat`), and create an xy-scatter diagram showing the true values x_n against the measurements z_n.
2. Design the linear MMSE estimator $\hat{x}_{lMMSE}(z) = \alpha z$, i.e. determine α, using estimates of $\mathrm{E}[xz]$ and $\mathrm{E}[z^2]$ derived from the first data set. Draw this estimator as a plot of $\hat{x}_{lMMSE}(z)$ versus z in the figure created in question 1.
3. Design the unbiased linear MMSE estimator $\hat{x}_{ulMMSE}(z) = \alpha z + \beta$, i.e. determine α and β, using estimates of $\mathrm{E}[x]$, $\mathrm{E}[z]$, $\mathrm{Cov}[x,z]$ and $\mathrm{Var}[z]$ derived from the first data set. Draw this estimator as a plot of $\hat{x}_{ulMMSE}(z)$ versus z in the figure created in question 1.
4. Load the second data set (`depthgauge_data_set2.mat`). Apply the linear MMSE estimator that you have developed with the first set to the measurements z_n from the second set. Determine the estimation errors $\tilde{x}_n = \hat{x}_{lMMSE}(z_n) - x_n$. Estimate the overall bias (mean of the estimation errors) and the variance of the errors.
5. Do the same with the unbiased linear MMSE estimator.
6. Explain the results, and answer the question: "why was it needed to use a second dataset?".

Hints:

- Estimators for the second order moments are:

$$M_{xz} = \mathrm{E}[xz] \quad \Rightarrow \quad \hat{M}_{xz} = \frac{1}{N}\sum_{n=1}^{N} x_n z_n$$

$$M_{zz} = \mathrm{E}[z^2] \quad \Rightarrow \quad \hat{M}_{zz} = \frac{1}{N}\sum_{n=1}^{N} z_n^2$$

$$\mathrm{Cov}[x,z] = E[(x-\overline{x})(z-\overline{z})] \quad \Rightarrow \quad \widehat{\mathrm{Cov}}[x,z] = \frac{1}{N-1}\sum_{n=1}^{N}(x_n-\hat{\overline{x}})(z_n-\hat{\overline{z}})$$

- Useful Matlab functions are `mean`, `var`, `cov` and `sum`.

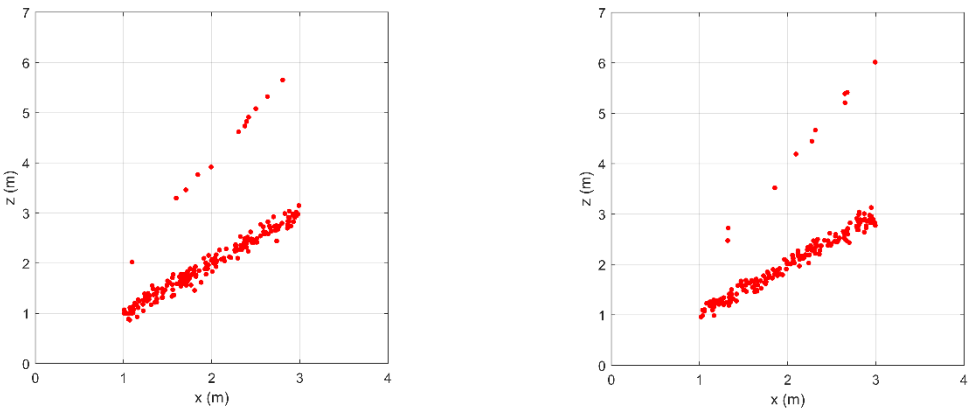


Figure 3 Two data sets

Exercise 3: Fundamentals of parameter estimation - part III

random vectors and unbiased linear MMSE estimation

Goal:

The objectives of this exercise are:
- to give the student insight about the concept of covariance matrices
- to provide the student knowledge about unbiased linear MMSE estimation

Prior knowledge:
We consider the navigation problem that is introduced in the first lecture. The parameter vector to estimate is the position $\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$ of a ship. The prior knowledge, obtained via dead reckoning, is captured as a (prior) expectation $\boldsymbol{\mu}_{\mathbf{x}}$ and a covariance matrix $\mathbf{C}_{\mathbf{x}}$. The latter expresses the (prior) uncertainty that we have about the position.

Measurement:
In order to increase the accuracy, the navigator of the ship takes a bearing. That is, he measures the direction φ of a beacon, e.g. a lighthouse, relative to the ship. See Figure 4. The beacon has a known reference position $\mathbf{x}_0$. Suppose that the compass reading is $\theta = \varphi + \Delta\theta$ with $\Delta\theta$ the (unknown) reading error. The line of sight is a geometrical representation of the measurement. It is the line that is defined by the position of the beacon and by the measured direction θ. The following equation defines the line of sight in the (ξ, η) plane:

$$x_0 \sin\theta - y_0 \cos\theta = \xi \sin\theta - \eta \cos\theta \tag{4}$$

The measurement model:
The relation between the ship's true position $\mathbf{x} = (x, y)$ and the true bearing φ is:

$$\begin{aligned} x_0 \sin\varphi - y_0 \cos\varphi &= x \sin\varphi - y \cos\varphi \qquad \text{or:} \\ x_0 \sin(\theta - \Delta\theta) - y_0 \cos(\theta - \Delta\theta) &= x \sin(\theta - \Delta\theta) - y \cos(\theta - \Delta\theta) \end{aligned} \tag{5}$$

The relation between the ship's true position $\mathbf{x}$ and the observed direction θ is nonlinear. To get a linear approximation, we apply a truncated Taylor series expansion to the sine and cosine functions:

$$\begin{aligned} \sin(\theta - \Delta\theta) &\approx \sin\theta - \Delta\theta \cos\theta \\ \cos(\theta - \Delta\theta) &\approx \cos\theta + \Delta\theta \sin\theta \\ &\Downarrow \\ x_0(\sin\theta - \Delta\theta\cos\theta) - y_0(\cos\theta + \Delta\theta\sin\theta) &\approx x(\sin\theta - \Delta\theta\cos\theta) - y(\cos\theta + \Delta\theta\sin\theta) \end{aligned} \tag{6}$$

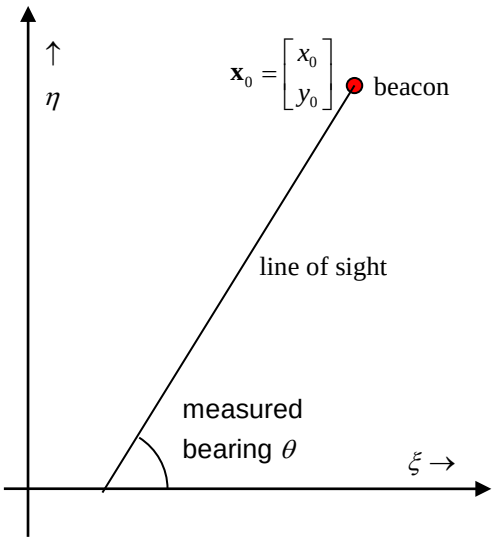


Figure 4 The bearing.
The line of sight is the line that is defined by the measured bearing and the position of the beacon

Rearrangement of terms yields:

$$x_0 \sin\theta - y_0 \cos\theta \approx x \sin\theta - y \cos\theta + \Delta\theta\big((x_0 - x)\cos\theta + (y_0 - y)\sin\theta\big) \tag{7}$$

Since $\theta \approx \varphi$, the factor $(x_0 - x)\cos\theta + (y_0 - y)\sin\theta$ almost equals $(x_0 - x)\cos\varphi + (y_0 - y)\sin\varphi$. The latter equals the distance d between beacon and ship. Therefore:

$$x_0 \sin\theta - y_0 \cos\theta \approx x \sin\theta - y \cos\theta + d\,\Delta\theta \tag{8}$$

This can be written in the form:

$$z = \mathbf{H}\mathbf{x} + v \qquad \text{with the following definitions:} \quad \begin{cases} z = x_0 \sin\theta - y_0 \cos\theta \\ \mathbf{H} = \begin{bmatrix} \sin\theta & -\cos\theta \end{bmatrix} \\ v = d\,\Delta\theta \end{cases} \tag{9}$$

Note that z and $\mathbf{H}$ depends on the position $\mathbf{x}_0$ of the beacon, and on the measured bearing θ. Both are known, and, therefore, z and $\mathbf{H}$ are also known. Although θ is the real measurement, we can use z as a derived measurement instead. The distance d is unknown, but can be estimated from prior knowledge $\boldsymbol{\mu}_\mathbf{x}$ of the ship's position and the position of the beacon: $d \approx \|\mathbf{x}_0 - \boldsymbol{\mu}_\mathbf{x}\|$. Assuming the measurement of the bearing has an uncertainty of $\sigma_{\Delta\theta}$, the standard deviation of v is $\sigma_v \approx d\,\sigma_{\Delta\theta}$. provided that $\sigma_{\Delta\theta}$ is expressed in radians.

The case

We consider the following case:

$$\text{prior knowledge:} \quad \boldsymbol{\mu}_\mathbf{x} = \begin{bmatrix} 10 \\ 20 \end{bmatrix} \qquad \mathbf{C}_\mathbf{x} = \begin{bmatrix} 25 & -25 \\ -25 & 70 \end{bmatrix}$$

$$\text{measurement:} \quad \mathbf{x}_0 = \begin{bmatrix} 100 \\ 100 \end{bmatrix} \qquad \theta = 35^0 \qquad \sigma_{\Delta\theta} = 1^0$$

Physical units are Nautical miles (Nm).

Uncertainty regions and principal axes:

The navigator wants to visualize the uncertainty region of the prior knowledge in a graph. This region is defined as the area within a contour (curve of equal height) of the pdf of $\mathbf{x}$. If $p(\mathbf{x})$ is the prior pdf of $\mathbf{x}$, then the equation $p(\mathbf{x}) = \text{constant}$ defines such a contour. For normal distributions,

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N |\mathbf{C}_\mathbf{x}|}} \exp\left(\frac{-(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^T \mathbf{C}_\mathbf{x}^{-1} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})}{2} \right),$$

the equation for the contour simplifies to:

$$(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^T \mathbf{C}_\mathbf{x}^{-1} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) = k^2 \text{ with } k = 1, 2 \text{ or } 3 \tag{10}$$

The position of the uncertainty region is defined by $\boldsymbol{\mu}_\mathbf{x}$. The shape and size are defined by the covariance matrix $\mathbf{C}_\mathbf{x}$. The contours, defined in (10), correspond to the 1σ-contour, the 2σ-contour or the 3σ-contour. Often, $k = 1$ is selected.

Eq. (10) defines an ellipse as the reader may verify by expanding the expression. See Figure 5. The shape and size of the ellipse is fully defined by two principal axes. In Figure 5, the directions of the axes are given by two vectors $\mathbf{v}_0$ and $\mathbf{v}_1$. The vectors have unit length, and are orthogonal. The lengths of the axes are given by the two scaling factors a_0 and a_1.

It is quite easy to find the principal axes of a covariance matrix: they are defined by its eigenvalues and corresponding eigenvectors. Eigenvectors and eigenvalues are the solutions from the equation:

$$\mathbf{C}_\mathbf{x}\mathbf{v} = \lambda\mathbf{v} \tag{11}$$

If the matrix $\mathbf{C}_\mathbf{x}$ is $M \times M$, then M solutions $\mathbf{v}_0, \cdots, \mathbf{v}_{M-1}$ exist with corresponding eigenvalues $\lambda_0, \cdots, \lambda_{M-1}$. The eigenvectors have unit length and are orthogonal. They point in the direction of the principal axes. The corresponding scaling factors appear to be $a_m = \sqrt{\lambda_m}$. For a proof, see Appendix C.3.1 and C.3.2 of the text book.

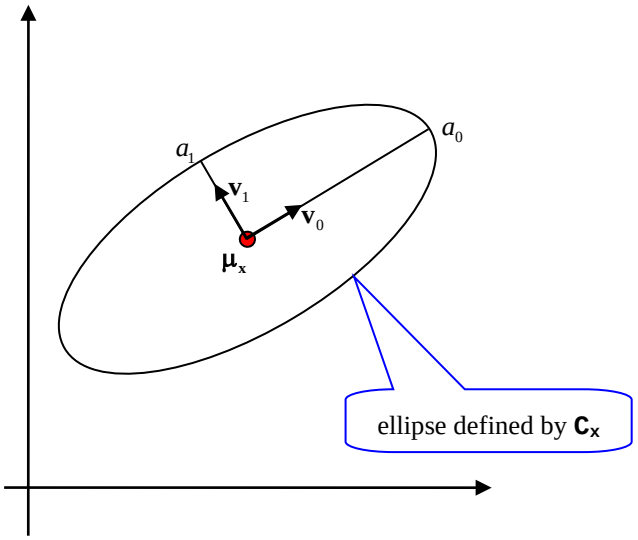


Figure 5 The principal axes of an ellipse

Questions:

1. Determine the eigenvalues and eigenvectors of $\mathbf{C}_\mathbf{x}$ (Matlab function: `eig`) and create Matlab code to draw the associated uncertainty region. This is done most conveniently by the following procedure described in pseudo code. The background of this procedure can be traced back in Appendix C.3.2 of the book.
   a. Generate a set of points on a circle with unit radius. The centre of the circle is positioned at the origin.
   b. Scale the x- and y coordinates of these points in accordance with the scaling factors a_0 and a_1. The resulting points form an ellipse with the right shape, but not with the right orientation and position.
   c. Rotate the set of points in accordance with the direction of the principal axes. This is most easily accomplished using the eigenvector-matrix returned by `eig`. This eigenvector-matrix is a rotation matrix.
   d. Shift the whole set to the position determined by $\boldsymbol{\mu}_\mathbf{x}$.
   e. Plot the curve defined by the resulting set of points.

   Implement this procedure as a Matlab-function, and let Matlab plot the uncertainty region. Make sure that the axis scales are equal (Matlab: **`axis equal`**). You will re-use this function later for several times. Put a graph of the result in your report.
2. With the given parameters of the measurement, add the line of sight to your figure. The uncertainty of the measured bearing is the standard deviation $\sigma_{\Delta\theta}$. The range $[\theta-\sigma_{\Delta\theta}, \theta+\sigma_{\Delta\theta}]$ defines an uncertainty region in the shape of a 2D cone. Visualize this cone in the graph by adding two dashed lines.
3. Our linearized measurement function replaces this cone by a bar, i.e. two parallel lines. The width of this bar is $2\sigma_v$. Calculate this width, and check visually whether the result makes sense.
4. Determine the derived measurement z, the measurement matrix $\mathbf{H}$, and the Kalman gain matrix. The covariance matrix of the measurement noise is $\mathbf{C}_\mathbf{v} = \sigma_v^2$. Next, calculate the unbiased linear MMSE estimate of the position and the corresponding (error) covariance matrix.
5. Draw the uncertainty region of the estimate. That is, plot the posterior mean and covariance matrix. Plot it in the same figure as in question 1. Explain.
6. Repeat question 2 up to 4 a number of times, but with varying values of $\sigma_{\Delta\theta}$, and explain what you see.
7. Repeat question 2 up to 4 a number of times, but with varying $\mathbf{C}_\mathbf{x}$. (Replace $\mathbf{C}_\mathbf{x}$ by $\alpha\mathbf{C}_\mathbf{x}$ with different values of α). Explain.

Exercise 4: propagation of uncertainty; prediction

dead reckoning

Goal:

The objectives of this exercise are:
- to give the student insight about the propagation of mean and covariance matrix in dynamic systems
- to learn the student how to use this insight for prediction

Introduction

Dead reckoning is a prediction technique that is an important tool in classical maritime navigation. The navigator uses a log to measure the speed of the vessel, and a compass to determine the heading (direction of the path). Together these two measurements provide information about the velocity $\mathbf{v}$. Suppose that $\hat{\boldsymbol{\xi}}(i)$ is an estimate that is available for the position $\boldsymbol{\xi}(i)$ at time i, and that $\mathbf{v}(i)$ is the velocity of the vessel determined at that time. Taken together, $\mathbf{v}(i)$ and $\hat{\boldsymbol{\xi}}(i)$ provide sufficient information to deduce an estimate of the position $\hat{\boldsymbol{\xi}}(i+\ell)$ valid for time $i+\ell$. Since $\hat{\boldsymbol{\xi}}(i+\ell)$ is based on measurements taken at time i, $\hat{\boldsymbol{\xi}}(i+\ell)$ can be regarded as a prediction with a lead of ℓ, i.e. a ℓ-step ahead prediction. The navigators not only provided a prediction of the position, but also a region of uncertainty (usually they draw a circle, an ellipse, or a diamond on a map around the predicted position).

In this exercise, we will implement a similar technique that is used nowadays as part in radar tracking applications. Using a state space description of the kinematics of a ship, we will develop a predictor that can predict the position of the ship and provides us with an uncertainty region. For that purpose, the mean and the covariance matrix of the state will be propagated in time.

But before we can actually implement the prediction we must find a suitable state space model that can describe the process. For that purpose, we use a registration of the path of ship.

System identification

The first task is to find a suitable state space model for the model. Figure 6 shows a registration of the position $\boldsymbol{\xi}(i) = [x(i) \quad y(i)]^T$ of the ship. We will use this registration to find a model. We will confine ourselves to a linear model of the type:

$$\mathbf{x}(i+1) = \mathbf{F}\mathbf{x}(i) + \mathbf{w}(i) \tag{12}$$

where $\mathbf{F}$ is a time invariant system matrix, and $\mathbf{w}(i)$ is a white sequence of Gaussian random vectors with zero mean and a time invariant covariance matrix $\mathbf{C}_{\mathbf{w}}$.

Particularly, we adopt the following kinematic model:

$$\begin{aligned} &\text{velocity:} && \mathbf{v}(i) = \boldsymbol{\xi}(i+1) - \boldsymbol{\xi}(i) \\ &\text{acceleration:} && \mathbf{a}(i) = \mathbf{v}(i+1) - \mathbf{v}(i) \\ & && \mathbf{a}(i+1) = \mathbf{F}_1\mathbf{a}(i) + \mathbf{w}_1(i) \end{aligned} \tag{13}$$

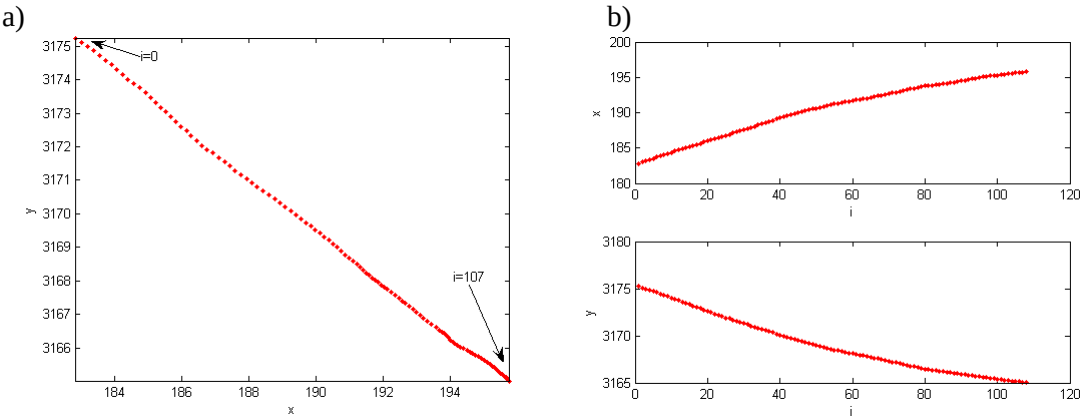


Figure 6 The path of a ship

The latter equation shows that the acceleration is modelled as a first order auto-regressive system. From that the velocity is obtained via integration of the acceleration, $\mathbf{v}(i+1)=\mathbf{v}(i)+\mathbf{a}(i)$. The position is obtained via integration of the velocity, $\boldsymbol{\xi}(i+1)=\boldsymbol{\xi}(i)+\mathbf{v}(i)$. Figure 7 shows $\mathbf{v}(i)$ and $\mathbf{a}(i)$ calculated from the registration of $\boldsymbol{\xi}(i)$. It can be seen that $\mathbf{a}(i)$ looks stationary. The results of a system identification based on the registration are:

$$\mathbf{F}_1 \cong \begin{bmatrix} -0.0595 & -0.1530 \\ -0.0813 & -0.1716 \end{bmatrix} \text{ and } \mathbf{C}_{\mathbf{w}_1} \cong \begin{bmatrix} 0.1177\times10^{-3} & -0.0026\times10^{-3} \\ -0.0026\times10^{-3} & 0.0782\times10^{-3} \end{bmatrix} \tag{14}$$

The state vector is defined as:

$$\mathbf{x}(i)=\begin{bmatrix} \boldsymbol{\xi}(i) \\ \mathbf{v}(i) \\ \mathbf{a}(i) \end{bmatrix} \tag{15}$$

1. Using the model of eq. (12), and the definitions of eq. (13), determine the matrix $\mathbf{F}$ and the covariance matrix $\mathbf{C}_{\mathbf{w}}$. Load the data set `log.mat`. A registration of $\boldsymbol{\xi}(i)$ with $i=0,\cdots,N-1$ is stored in a $2\times N$ dimensional array `xsi`. In this particular case, $N=108$. Calculate a $2\times(N-1)$ array `v` that contains the differences $\mathbf{v}(i)=\boldsymbol{\xi}(i+1)-\boldsymbol{\xi}(i)$, and a $2\times(N-2)$ array a that contains the differences $\mathbf{a}(i)=\mathbf{v}(i+1)-\mathbf{v}(i)$. Use the arrays `xsi`, `v` and `a` to create a $6\times(N-2)$ array `x` that contain a registration of the state vector $\mathbf{x}(i)$.
2. Suppose that the current time[2] is $i=10$, and that the current state $\mathbf{x}(10)$ is precisely known. (You can retrieve this state from the array `x` created in question 1). We want to predict the position at the time $j=i+90=100$. That is, using $\mathbf{x}(10)$ we want to predict the position $\ell=90$ steps ahead. The prediction of $\mathbf{x}(j)$, knowing $\mathbf{x}(i)$, is denoted by $\hat{\mathbf{x}}(j\,|\,i)$. The prediction just equals the expectation of $\mathbf{x}(j)$ given the state $\mathbf{x}(i)$. Thus, $\hat{\mathbf{x}}(j\,|\,i)=\mathrm{E}[\mathbf{x}(j)\,|\,\mathbf{x}(i)]$. Create Matlab code that calculates the predicted path from time $j=11$ up to time $j=100$. That is, the sequence $\hat{\mathbf{x}}(i+\ell\,|\,i)$ with $\ell=1,2,\cdots,90$.
3. Create the plot shown in Figure 6a. Here, $y(i)$ (Matlab: `x(2,:)`) is plotted against $x(i)$ (Matlab: `x(1,:)`) for $i=0,\cdots,j$. Draw in the same figure the predicted path calculated in question 2.
4. Since in question 2 the knowledge of $\mathbf{x}(10)$ is exact, the covariance matrix for that time is zero: $\mathbf{C}_{\mathbf{x}}(i)=\mathbf{0}$. Suppose that the covariance matrix of a prediction $\hat{\mathbf{x}}(j\,|\,i)$ is denoted by $\mathbf{C}(j\,|\,i)$. (This covariance matrix is also the covariance matrix of the prediction error $\hat{\mathbf{x}}(j\,|\,i)-\mathbf{x}(j)$). Create Matlab code that calculates the sequence of covariance matrices, i.e. $\mathbf{C}(i+\ell\,|\,i)$ with $\ell=1,2,\cdots,90$. The covariance matrix of $\hat{\boldsymbol{\xi}}(j\,|\,i)$ is a submatrix from $\mathbf{C}(j\,|\,i)$. Draw the ellipse associated with this submatrix. Draw this ellipse centred at the position $\hat{\boldsymbol{\xi}}(j\,|\,i)$. Do this in the same figure as the previous plot.
5. Repeat question 2, 3, and 4 with the following values of i: 30, 50, 70 and 90. (j is fixed to 100; so the lead is 70, 50, 30, and 10, respectively). Explain the results.

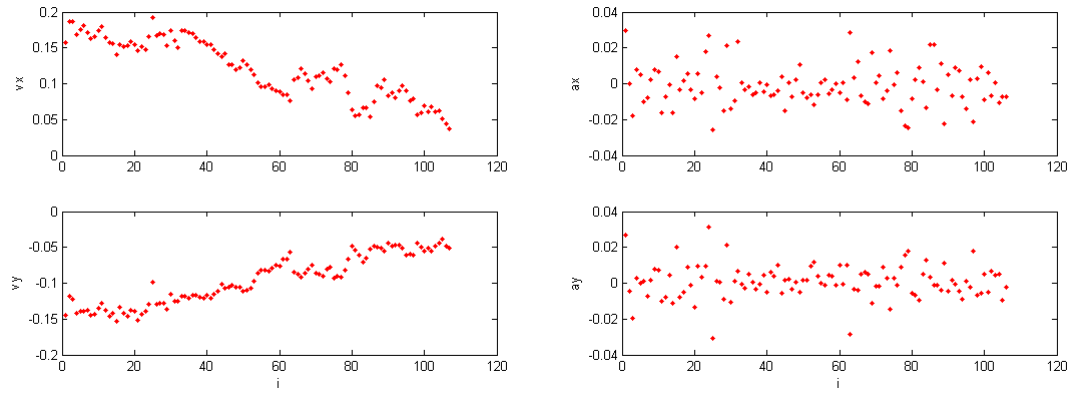


Figure 7 The velocity and the acceleration

[2] The time index i starts at zero. But, since Matlab starts indexing an array always at 1, $\boldsymbol{\xi}(i)$ is stored in `xsi(*,i+1)`.

Exercise 5: discrete kalman filtering

radar tracking

Goal:

In this exercise, the student learns how to implement a discrete Kalman filter.

Introduction

We consider the same dynamic process as in exercise 4: a kinematic model that is used for tracking. The state vector consists of the 2-D position $\boldsymbol{\xi}(i)$, the velocity $\mathbf{v}(i)$, and the acceleration $\mathbf{a}(i)$ of a vehicle. The acceleration is modelled as a stable, first order AR process:

$$\mathbf{x}(i+1) = \mathbf{F}\mathbf{x}(i) + \mathbf{w}(i) \quad \text{with} \quad \mathbf{x}(i) = \begin{bmatrix} \boldsymbol{\xi}(i) \\ \mathbf{v}(i) \\ \mathbf{a}(i) \end{bmatrix}; \quad \mathbf{F} \text{ and } \mathbf{w}(i) \text{ defined by:} \quad \begin{aligned} \boldsymbol{\xi}(i+1) &= \boldsymbol{\xi}(i) + \mathbf{v}(i) \\ \mathbf{v}(i+1) &= \mathbf{v}(i) + \mathbf{a}(i) \\ \mathbf{a}(i+1) &= \mathbf{F}_1\mathbf{a}(i) + \mathbf{w}_1(i) \end{aligned} \tag{16}$$

$\mathbf{F}_1$ is the system matrix for $\mathbf{a}(i)$. $\mathbf{w}_1(i)$ is the process noise of the acceleration. $\mathbf{C}_{\mathbf{w}_1}$ is the covariance matrix of $\mathbf{w}_1(i)$. $\mathbf{F}_1$ and $\mathbf{C}_{\mathbf{w}_1}$ are given by:

$$\mathbf{F}_1 \cong \begin{bmatrix} 0.97 & 0 \\ 0 & 0.97 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_{\mathbf{w}_1} \cong \begin{bmatrix} 0.0016 & 0 \\ 0 & 0.0016 \end{bmatrix} \tag{17}$$

The position of the vehicle is measured by a radar system. See Figure 8. The radar system is modelled by:

$$\mathbf{z}(i) = \boldsymbol{\xi}(i) + \mathbf{n}(i) \tag{18}$$

$\mathbf{n}(i)$ is the measurement noise with assumed covariance matrix:

$$\mathbf{C}_\mathbf{n} = \begin{bmatrix} 49 & 0 \\ 0 & 49 \end{bmatrix} \tag{19}$$

We deviate from the convention that measurement noise is denoted by $\mathbf{v}(i)$ since this symbol is already used for velocity.

There is not much prior knowledge of $\mathbf{x}(0)$. The mean value of $\mathbf{x}(0)$ is zero. The position $\boldsymbol{\xi}(0)$ has an uncertain range of 100 (standard deviation). The velocity $\mathbf{v}(0)$ has an uncertain range of 4. The acceleration $\mathbf{a}(0)$ has an uncertain range of about 0.2.

1. Determine the matrices $\mathbf{F}$, $\mathbf{C}_\mathbf{w}$ and $\mathbf{C}(0|-1) \overset{def}{=} \mathbf{C}_\mathbf{x}(0)$. The latter is the covariance matrix of the initial pdf $p(\mathbf{x}(0))$. Determine also $\hat{\mathbf{x}}(0|-1) = \mathrm{E}[\mathbf{x}(0)]$, i.e. the expectation of the initial pdf.
2. Load the matrix `zradar.mat`. This file contains the measurements $\mathbf{z}(i)$ with $i = 0, \cdots, 99$ stored in an

a) b)

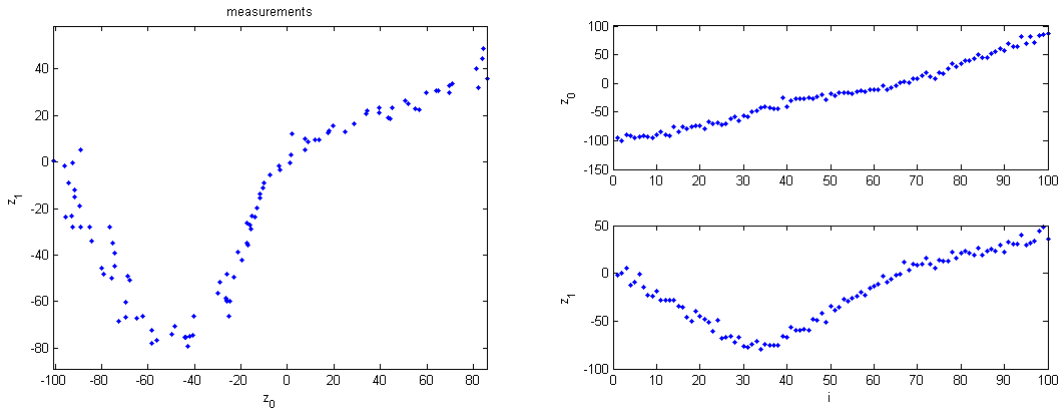


Figure 8 The measurements obtained from a radar target.

2×100 -array `z`. Create a for-loop with the counter `i` running from[3] 0 up to 99. Within the loop you implement Matlab code that keeps track of the estimate $\hat{\mathbf{x}}(i\,|\,i)$ and the corresponding error covariance matrix $\mathbf{C}(i\,|\,i)$. You do so by also keeping track of the prediction $\hat{\mathbf{x}}(i\,|\,i-1)$ and the corresponding prediction covariance matrix $\mathbf{C}(i\,|\,i-1)$. So, the inner part of the loop consists of:

- An update:
  Using the current measurement $\mathbf{z}(i)$, and the predicted state $\hat{\mathbf{x}}(i\,|\,i-1)$ with corresponding covariance matrix $\mathbf{C}(i\,|\,i-1)$, you calculate the current estimate $\hat{\mathbf{x}}(i\,|\,i)$ with corresponding error covariance matrix $\mathbf{C}(i\,|\,i)$.
- A prediction:
  Using the current estimate $\hat{\mathbf{x}}(i\,|\,i)$ with corresponding error covariance matrix $\mathbf{C}(i\,|\,i)$, you calculate a one-step ahead prediction $\hat{\mathbf{x}}(i+1\,|\,i)$ with corresponding covariance matrix $\mathbf{C}(i+1\,|\,i)$.

Keep a log of the vectors $\hat{\mathbf{x}}(i\,|\,i)$ and $\hat{\mathbf{x}}(i\,|\,i-1)$ and the matrices $\mathbf{C}(i\,|\,i)$ and $\mathbf{C}(i\,|\,i-1)$. One possibility is to store them in cell-arrays (Matlab: `Cerrlist{i} = Cerr`, and so on).

3. Create a plot with the measurements as in Figure 8a. Using the log created in question 3, plot in the same figure (with different colour) the estimated positions, and the predicted positions. Make sure that the axis scales are equal. Which of the three sequences looks most noisy, and which most smooth? Explain!
4. Using the log in question 3, add the uncertainty regions (ellipses) to the estimated positions for the following time indices $i = 0,3,6,\cdots$ (Matlab: `for i=1:3:100` $\cdots$). What do you observe with respect to the shape and size of the ellipse depending in i? Explain.
5. The Matlab control toolbox contains a function `dlqe` that enables you to calculate the Kalman gain matrix, and the covariance matrices of the steady state solution.

   Matlab: `[K,Cpred,Cerr] = dlqe(F,eye(6),H,Cw,Cn).`

   Execute this command, and compare the matrices `K,Cerr,Cpred` with your results. To what extend do they agree? Rerun the Kalman filter with the steady state matrices instead of the time-variant one, i.e. repeat question 2 up to 4, but this time with `K,Cerr,Cpred` derived from `dlqe`. Compare and explain.

[3] In fact, it may be easier to let `i` run from 1 up to 100, because in Matlab the array indices always start at 1, so that $\mathbf{z}(i)$ is stored in `z(*,i+1)`.

Exercise 6: extended Kalman filtering

Dead reckoning and bearing measurements

Introduction

In exercise 3, we applied the unbiased linear MMSE estimation to update the position of a ship using a line-of-sight measurement knowing the position of an observed beacon. In exercise 4 we have learnt how to predict the position, velocity, and acceleration of a ship. Exercise 5 combined the two techniques. In the current exercise, this is done again, but in contrast with the previous exercises the system equation and the measurement function will not be linearized beforehand, but during tracking: we implement an extended Kalman filter.

The system equation of a yacht

The motion of a ship is a balance between forces and moments. The basic equation is:

$$\underbrace{\boldsymbol{M}\dot{\boldsymbol{\nu}}}_{\text{inertia}} + \underbrace{\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{resistance}} + \underbrace{\boldsymbol{g}(\boldsymbol{\xi})}_{\text{weight and buoyance}} = \underbrace{\boldsymbol{\tau}}_{\text{external forces}} \tag{20}$$

The 6-dimensional vector $\boldsymbol{\nu}$ contains the linear velocity and the angular velocity (turn rates). $\boldsymbol{\xi}$ is the 3-dimensional position. Eq (20) looks simple, but in fact, if the formula is worked out, each individual term appears to be quite complicated. We therefore will assume a much simpler model. Specifically, we will ignore:

- orientation and rotations of the ship
- the variations in altitude (heaving)
- a number of external forces

Instead of 6-dimensional, $\boldsymbol{\nu}$ becomes 2-dimensional containing only the linear velocity in x- and y-direction, that is $\mathbf{v} = [v_x \quad v_y]^T$. The ignorance of rotations implies that Coriolis forces and centripetal forces are ruled out. The forces due to weight and buoyancy (Archimedes) merely effect the altitude of the ship. So, the term $\boldsymbol{g}(\boldsymbol{\xi})$ is ruled out too. This leaves us with the following model:

$$m\dot{\mathbf{v}} + d(\mathbf{v})\mathbf{v} = \mathbf{t} \tag{21}$$

m is the mass of the ship. $d(\mathbf{v})\mathbf{v}$ represents the resistance. $\mathbf{t}$ is the net external force. The factor $d(\mathbf{v})$ is the damping factor. Many physical aspects contribute to this factor. The most important ones are the skin friction (important especially for low speeds), and the wave making resistance (for higher speeds). We will model the damping factor according to the following approximation:

$$d(\mathbf{v}) = a|\mathbf{v}| + b|\mathbf{v}|^3 \tag{22}$$

The first term describes the skin friction. The second term reflects the damping due to wave making. **Figure 9** shows the resistance $d(\mathbf{v})|\mathbf{v}|$ of a ship. The red line shows an accurate model of the wave making resistance. The term $b|\mathbf{v}|^3$ in eq (22) is an approximation (the blue line) that ignores the fluctuating behaviour of the damping factor. The constants a and b are parameters that depend on the size and shape of the ship's hull.

The external force $\mathbf{t}$ is characterized by a heading ϕ (the direction), and a thrust t (the magnitude). For a sailing yacht, the thrust is caused by the wind. We neglect the influence of waves. For simplicity, we assume a first order differential equation:

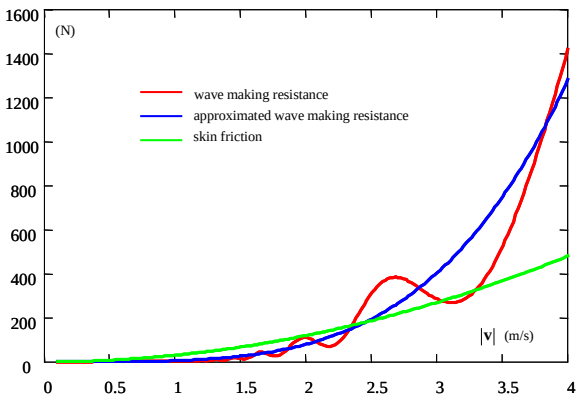


Figure 9 The resistance of a ship

$$\dot{t} = -\alpha_t\left(t - t_0\right) + \text{process noise} \tag{23}$$

t_0 is the mean thrust. The differential equation describes slow fluctuations of the wind. The factor $1/\alpha_t$ is a physical constant that characterizes the slowness of these fluctuations. For now, we assume that the intended heading ϕ_0 is constant. However, due to a variety of reasons the real heading ϕ randomly deviates from ϕ_0. Here, we also assume a first order differential equation:

$$\dot{\phi} = -\alpha_\phi(\phi - \phi_0) + \text{process noise} \tag{24}$$

The factor $1/\alpha_\phi$ is a time constant that describes how fast deviations of the heading are corrected.

Time discrete system model

Suppose that the sampling period is Δ. Then, the time derivatives are roughly approximated by:

$$\dot{\boldsymbol{\xi}} = \mathbf{v}(i) \cong \frac{\boldsymbol{\xi}(i+1)-\boldsymbol{\xi}(i)}{\Delta} \qquad \mathbf{a}(i)=\dot{\mathbf{v}} \cong \frac{\mathbf{v}(i+1)-\mathbf{v}(i)}{\Delta} \qquad \dot{t} \cong \frac{t(i+1)-t(i)}{\Delta} \qquad \dot{\phi} \cong \frac{\phi(i+1)-\phi(i)}{\Delta} \tag{25}$$

The following equations make up our model of the process:

$$\begin{aligned}
&\boldsymbol{\xi}(i+1) = \boldsymbol{\xi}(i) + \Delta\mathbf{v}(i) + \mathbf{w}_{\boldsymbol{\xi}}(i) && \text{the 2-dimensional position}\\
&\mathbf{v}(i+1) = \mathbf{v}(i) + \Delta\mathbf{a}(i) + \mathbf{w}_{\mathbf{v}}(i) && \text{the 2-dimensional velocity}\\
&\mathbf{a}(i+1) = \frac{1}{m}\left(t(i)\begin{bmatrix}\cos\left(\phi(i)\pi/180\right)\\ \sin\left(\phi(i)\pi/180\right)\end{bmatrix} - d\left(\mathbf{v}(i)\right)\mathbf{v}(i)\right) + \mathbf{w}_{\mathbf{a}}(i) && \text{the 2-dimensional acceleration}\\
&t(i+1) = t(i) - \Delta\alpha_t\left(t(i)-t_0\right) + w_t(i) && \text{the 1-dimensional thrust (magnitude of the force)}\\
&\phi(i+1) = \phi(i) - \Delta\alpha_\phi\left(\phi(i)-\phi_0\right) + w_\phi(i) && \text{the heading (direction of the force)}
\end{aligned} \tag{26}$$

$w_\phi(i)$ is the process noise of the heading of the ship. Its variance is $\sigma^2_{w_\phi}$. $w_t(i)$ is the process noise of the thrust. Its variance is $\sigma^2_{w_t}$. The 2-dimensional vectors $\mathbf{w}_{\boldsymbol{\xi}}(i)$, $\mathbf{w}_{\mathbf{v}}(i)$, and $\mathbf{w}_{\mathbf{a}}(i)$ are also process noise terms. They account for small unpredictable errors in the modelled position, velocity, and acceleration. These small errors might be caused by the approximation of the time derivatives by the time differences as indicated in eq. (25). These errors are assumed to be uncorrelated. Their standard deviations are $\sigma_{w_{\boldsymbol{\xi}}}$, σ_{w_v} and σ_{w_a}.

For a particular yacht, and a particular weather situation, the following parameters apply:

$\sigma_{w_{\boldsymbol{\xi}}} = 10^{-2}$	m	standard deviation process noise on position
$\sigma_{w_v} = 10^{-2}$	m/s	standard deviation process noise on velocity
$\sigma_{w_a} = 10^{-2}$	m/s^2	standard deviation process noise on acceleration
t_0=400	N	intended thrust
$\sigma_{w_t} = 8$	N	standard deviation process noise on thrust
ϕ_0=45°		intended heading
$\sigma_{w_\phi} = 0.5°$		standard deviation process noise on heading
$\alpha_t = \frac{1}{1200}$	s^{-1}	coefficient differential equation for thrust
$\alpha_\phi = \frac{1}{200}$	s^{-1}	coefficient differential equation for heading
$\Delta = 1$	s	sampling period
$m = 3600$	kg	mass
$a = 30$	kg/m	skin friction constant
$b = 5$	kg s^2/m^3	wave making constant

(27)

The state vector $\mathbf{x}(i)$ is 8-dimensional and comprises the position $\boldsymbol{\xi}(i)$, the velocity $\mathbf{v}(i)$, the acceleration $\mathbf{a}(i)$, the thrust $t(i)$, and the heading $\phi(i)$. With that, the system function in eq (26) can be abbreviated to:

$$\mathbf{x}(i+1) = \mathbf{f}\left(\mathbf{x}(i),\mathbf{u}(i)\right) + \mathbf{w}(i) \qquad \text{with} \qquad \mathbf{u}(i) = \begin{bmatrix} t_0(i)\\ \phi_0(i)\end{bmatrix} \tag{28}$$

The measurement model

Measurements are done with a sampling period 500 s. Therefore, the data rate of the measurements is much lower than the sampling rate of the time discrete system model. The discrete moments at which measurements are available are: $i_m = 0, 500, 1000, \cdots$. The measurements themselves are denoted by $\mathbf{z}\left(i_m\right)$.

The ship is provided with three different types of sensors:

- A beacon-based measurement.
- A log which measures the speed of the ship.
- A compass that measures the heading direction (direction of movement) of the ship.

The measurement function is given by:

$$\mathbf{z}(i_m) = \mathbf{h}\left(\mathbf{x}(i_m)\right) + \mathbf{n}(i_m) = \begin{bmatrix} \dfrac{180^\circ}{\pi} \arctan\left(\dfrac{y_0 - \xi_y(i_m)}{x_0 - \xi_x(i_m)} \right) \\ \left|\mathbf{v}(i_m)\right| \\ \dfrac{180^\circ}{\pi} \arctan\left(\dfrac{v_y(i_m)}{v_x(i_m)} \right) \end{bmatrix} + \begin{bmatrix} n_1(i_m) \\ n_2(i_m) \\ n_3(i_m) \end{bmatrix} \tag{29}$$

The beacon-based measurement, $z_1(i_m)$ is the same as in exercise 3, but this time we do not linearize the equation in advance. The position of the beacon is at $\mathbf{x}_0 = \begin{bmatrix} 5000 \text{ m} & 10000 \text{ m} \end{bmatrix}^T$. The standard deviation of the reading error $n_1(i_m)$ of the compass is $\sigma_{n_1} = 1^\circ$. The speed measurement has a standard uncertainty of $\sigma_{n_2} = 0.3$ m/s. The standard deviation of the error in the measured heading is $\sigma_{n_3} = 1^\circ$. The three error sources are not correlated.

The prior knowledge at time $i = 0$ is as follows:

- The position $\boldsymbol{\xi}(0)$: $\mathrm{E}[\boldsymbol{\xi}(0)] = \mathbf{0}$ m and standard deviations $\sigma_{\xi_x} = \sigma_{\xi_y} = 10000$ m.
- The velocity $\mathbf{v}(0)$: $\mathrm{E}[\mathbf{v}(0)] = \mathbf{0}$ m/s and standard deviations $\sigma_{v_x} = \sigma_{v_y} = 2$ m/s.
- The acceleration $\mathbf{a}(0)$: $\mathrm{E}[\mathbf{a}(0)] = \mathbf{0}$ m/s^2 with standard deviations $\sigma_{a_x} = \sigma_{a_y} = 0.04$ m/s^2.
- The thrust: $t(0)$ $\mathrm{E}[t(0)] = 400$ N with standard deviation $\sigma_t = 300$ N.
- The heading: $\phi(0)$ $\mathrm{E}[\phi(0)] = 0^\circ$ with standard deviation $\sigma_\phi = 10^\circ$.

The measurement data is available in a data file `z_yacht.mat`. This file contains an 3×20-array `z` with the measurement data $\mathbf{z}(i_m)$. It also contains a 20-dimensional array `imeas` containing** the 20 discrete times i_m.

This exercise requires the availability of Matlab functions that implements the system function, the measurement function, and the associated Jacobian matrices. They are given in the following m-files:

`fsys.m`	the system function $\mathbf{f}(\mathbf{x}(i), \mathbf{u}(i))$	the control input vector is: $\mathbf{u}(i) = \begin{bmatrix} t_0(i) & \phi_0(i) \end{bmatrix}^T$
`Fjacobian.m`	the Jacobian matrix of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ with respect to $\mathbf{x}$	
`hmeas.m`	the measurement function $\mathbf{h}(\mathbf{x})$	
`Hjacobian.m`	the Jacobian matrix of $\mathbf{h}(\mathbf{x})$ with respect to $\mathbf{x}$.	

Read the help of these functions carefully before using them. The functions `fsys` *and* `hmeas` *can be used with or without an optional third argument: "look before you leap".*

Questions:

1. The covariance matrix $\mathbf{C}_\mathbf{w}$ of the process noise is an 8×8 matrix. See the text above. Create this matrix in Matlab. Likewise, create the covariance matrix $\mathbf{C}_\mathbf{v}$ of the measurement noise.
2. Load the file `z_yacht.mat`. Implement the extended Kalman filter for this problem. Run the code starting with time index $i = 0$ (Matlab: `i=1`), and ending with time index $i = 9999$. See††.
3. Create graphs of the estimated states (position, velocity, acceleration, and force versus time).
4. Create also a graph of the estimated path. Add, for some moments of time, uncertainty regions to this graph. Make sure that the axis scales are equal in horizontal and vertical direction (`axis equal`).
5. Extend the code such that the NIS is calculated for each available measurement $\mathbf{z}(i_m)$. That is for each time index i_m. Perform a quick test whether these $nis(i_m)$ complies with the required probability density.
6. Are the results according to your expectations? Explain.

** Matlab arrays start indexing with 1 instead of 0. Therefore, the array is: `imeas=1,501,1001,...`

†† For this problem, most of the cycles of the Kalman filter consist of just predictions. Updates take place only at $i \equiv i_m$.

exercise 7: particle filtering

Introduction

In this exercise, the student implements a particle filtering, performs a consistency check, and compares the results with that from an extended Kalman filter.

Textbook:

- Section 4.4
- Section 9.3.3 (especially the part on consistency checks, and the Matlab implementation)

The physical process and sensor system

We consider the same physical process and sensor system as in exercise 6. There are two differences:

- In contrast to exercise 6, we now assume that the intended heading $\phi_0(i)$ is time variant. However, since the heading of the ship is the navigator's decision, and therefore known, we assume that $\phi_0(i)$ is given at any moment of time. We apply the model of eq. (26) launched in exercise 6, except that now ϕ_0 is not constant.
- The prior knowledge at time $i=0$ is the same as in exercise 6, except for the position. We now assume that $\sigma_{\xi_x} = \sigma_{\xi_y} = 100$ m instead of $\sigma_{\xi_x} = \sigma_{\xi_y} = 10000$ m.

Consistency checks for particle filtering

In [1], page 345, variables $u_n(i)$ are introduced that can be used for consistency checks. For each individual measurement element $z_n(i)$, we consider the probability density $p_n(z,i) \overset{def}{=} p\left(z_n(i) \mid \mathbf{Z}(i-1)\right)$, i.e. the pdf of $z_n(i)$ given all previous measurements. The cumulative distribution of $z_n(i)$ is given then by:

$$F_n(z,i) \overset{def}{=} \Pr\left(z_n(i) \le z \mid \mathbf{Z}(i)\right) = \int_{-\infty}^{z} p_n(\zeta,i)\,d\zeta \tag{1}$$

Using this cumulative distribution, we form test variables $u_n(i) \overset{def}{=} F_n\left(z_n(i) \mid i\right)$ for each measurement element. It can be proven that if the particle filter behaves consistently, then the variables $u_n(i)$ are uniformly distributed between 0 and 1. Formal statistical procedures are available to test whether a collection of variables is uniformly distributed (see [1], page 346), but we confine ourselves to a quick impression using plots of the test variables (as in [1], Figure 9.16 and Figure 9.18).

Another variable of interest is the so-called effective number of particles:

$$K_{eff}(i) = \frac{1}{\sum_{k=1}^{K}\left(w_{norm}^{(k)}(i)\right)^2} \tag{2}$$

This variable can be used to see whether the normalized weight factors are not too much unevenly distributed. In the extreme case, we have $K_{eff} = 1$ indicating that only one weight factor is nonzero.

Questions:

1. Implement a particle filter for this problem, and apply it to the data in the file `z_yacht2.mat`. Using the particles, calculate the MMSE estimate $\hat{\mathbf{x}}_{MMSE}(i) = \overline{\mathbf{x}}(i \mid i)$. Create a plot of the estimated path.
2. Extend you code of the particle filtering such that for time at which a measurement is available the covariance matrix of the estimate is calculated, i.e. $\hat{\mathbf{C}}(i \mid i)$. Using these estimates, add plots of the uncertainty areas to the figure of question 1. Interpret the result.
3. Extend the code such that the test variables $u_n(i_m)$ with $n = 1, \cdots, 3$ are calculated. Plot these test variables in one graph and use colours to distinguish them. That is, use the option `'r.'`, `'b.'` and `'g.'`. Interpret the result.
4. Extend the code such that the effective number of particles is calculated after each measurement. Plot this variable, and interpret the result.
5. Perform some experiments such that you are sure that you have a sufficient number of particles.
6. Adjust your code of the EKF from exercise 6, such as to anticipate time variant ϕ_0's. Run the code once

again. Interpret the results (plot with path + uncertainty regions, and the plot of the NIS).

7. Which of the two estimation methods do you prefer? Motivate! Computational load is only of secondary importance.

Hints:

1. You can recycle the code from Listing 9.10 (available via http://www.prtools.org). Use it as example-code only. You should replace the parts of the code that are specific for the application. Some changes that are needed are:
   - The definition of parameters of the physical process and the sensor system.
   - The initialization of the particles.
   - The generation of `Zs` (predicted measurement particles representing $p(\mathbf{z}(i) \mid \mathbf{Z}(i-1))$). You can use the function `hmeas` (exercise 6) together with the sensor noise model.
   - In the update step: the term `H*Xs` should be replaced by a call to `hmeas`.
   - In the prediction step: all lines can simply be replaced by a call to `fsys` (exercise 6).

   In addition, you can remove:
   - The first and last line (we don't need to implement the particle filter as a function; an m-file script suffices).
   - The part which calculates the histograms and the value of J. (We are not going to optimize parameters of the model).

   Apart from that you should also make a provision that skips the update for time steps without measurements (as in exercise 6).
2. Try to prevent `for-loops` as much as possible. `For-loops` dramatically slow down the execution time. The function `fsys` and `hmeas` are able to handle all particles in one call.
3. Be aware of the 'modulo' properties when dealing with angles. For instance, the difference between 179° and -179° is -2°, and not $+358^\circ$.

[1] F. van der Heijden et al: *Classification, Parameter Estimation and State Estimation*, Wiley&Sons, Chichester 2004.

Exercise 8: SLAM

Introduction

We consider a typical SLAM problem: Simultaneous Localization and Mapping:

- A vehicle is moving around in an unexplored area.
- The motion of the vehicle is monitored in terms of heading and speed.
- The track of the vehicle can be estimated by means of the heading and the speed, but errors will accumulate. Therefore, this estimate will become inaccurate in due course of time.
- The space is provided with a number of landmarks. These are identifiable and static points. Beforehand, the number of landmarks and their positions are fully unknown.
- The vehicle is equipped with a sensor system that is able to locate the positions of the landmarks relative to its own position.
- Only landmarks that are nearby the vehicle can be measured. That is, the landmarks should be within the visible range of the sensor system.

The purpose of SLAM is to improve the estimate of the track using the observations of the landmarks, and at the same time to estimate the (static) positions of these landmarks.

The system equation of the vehicle

The position of the vehicle at time i is denoted by $\mathbf{p}(i) = \begin{bmatrix} x(i) & y(i) \end{bmatrix}^T$. The speed of the vehicle is $v(i)$, and the heading is $\phi(i)$. We assume that these motion variables are measured at the beginning of each sampling period. Both come with an uncertainty due to measurement noise. The system equation of the vehicle then becomes:

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} x(i) \\ y(i) \end{bmatrix} + \Delta\left(v(i) + \tilde{v}(i)\right) \begin{bmatrix} \cos\left(\phi(i) + \tilde{\phi}(i)\right) \\ \sin\left(\phi(i) + \tilde{\phi}(i)\right) \end{bmatrix} \tag{1}$$

Δ is the sampling period. The random variables $\tilde{v}(i)$ and $\tilde{\phi}(i)$ represent the uncertainty due to the measurement noise. These variables also embed the uncertainty that arises from the time-discrete approximation of a time-continuous process. Their standard deviations are σ_v and σ_ϕ, respectively.

Note that the system equation (20) deals with $\phi(i)$ and $v(i)$ as if they are control variables. As such, $\tilde{v}(i)$ and $\tilde{\phi}(i)$ are regarded as process noise. The sequences $\tilde{v}(i)$ and $\tilde{\phi}(i)$ are considered to be uncorrelated in time (white noise sequences).

The control signals are available as vectors $\mathbf{u}(i) = \begin{bmatrix} v(i) & \phi(i) \end{bmatrix}^T$ in a matlab[‡‡] array `u`. The headings are given in degrees. Therefore, the cosine and sine in eq (1) and (2) must be provided with a factor $\pi/180$, e.g. $\cos\left(2\pi\phi/180\right)$

Part I: prediction only

Suppose that we want to estimate the track without using the landmarks. Such a prediction is based on a linear approximation of the system equation:

$$\mathbf{p}(i+1) = \mathbf{F}\mathbf{p}(i) + \underbrace{\Delta v(i) \begin{bmatrix} \cos\phi(i) \\ \sin\phi(i) \end{bmatrix}}_{\text{control}} + \underbrace{\mathbf{G}\left(\mathbf{u}(i)\right) \begin{bmatrix} \tilde{v}(i) \\ \tilde{\phi}(i) \end{bmatrix}}_{\text{process noise}} \tag{2}$$

$\mathbf{G}(\mathbf{u})$ is the Jacobian matrix of the system equation with respect to the process noise. The sampling period is $\Delta = 0.25$ s. The standard deviations are: $\sigma_v = 5$ m/s and $\sigma_\phi = 15^0$.

[‡‡] Matlab data is available in the file `SLAM.mat`

Questions:

1. Determine the system matrix $\mathbf{F}$.
2. The process noise $\mathbf{w}(i)$ is made up by the Jacobian matrix $\mathbf{G}(\mathbf{u})$ and the uncorrelated sequences $\tilde{v}(i)$ and $\tilde{\phi}(i)$. That is:

$$\mathbf{w}(i) = \mathbf{G}\left(\mathbf{u}(i)\right)\begin{bmatrix}\tilde{v}(i)\\ \tilde{\phi}(i)\end{bmatrix} \tag{3}$$

   Give the covariance matrix of $\begin{bmatrix}\tilde{v}(i) & \tilde{\phi}(i)\end{bmatrix}^T$. Denote this matrix by $\mathbf{C}_{\tilde{\mathbf{w}}}$. Give an expression of the Jacobian matrix $\mathbf{G}(\mathbf{u})$, and finally give the expression for the covariance matrix $\mathbf{C}_{\mathbf{w}}$ (this expression is to given only in terms of $\mathbf{G}(\mathbf{u})$ and $\mathbf{C}_{\tilde{\mathbf{w}}}$).
3. By definition, the system starts at a position $\mathbf{p}(0) = \mathbf{0}$ with no uncertainty. Develop a Matlab script that predicts the trajectory. Insert uncertainty regions in your graph at $i = 100, 200, \cdots$ (you can use the function `mod`).

Part II: bearing and distance measurements

In this part we assume that landmarks are available. They are only visible if they are within a range of about 22 m. The landmarks are static. This implies that their system function is without process noise: suppose that the location of the m-th landmark is given by the 2D vector $\mathbf{x}_m(i)$, then:

$$\mathbf{x}_m(i+1) = \mathbf{x}_m(i) \tag{4}$$

The sensor system measures the positions of the landmarks relative to its own position. A bearing and distance sensor, e.g. an optical range finder, measures the position of a landmark in a polar coordinate system that is attached to the vehicle. This can easily be cast into a Cartesian system. The result is that the m-th landmark is measured as:

$$\mathbf{z}_m(i) = \mathbf{x}_m(i) - \mathbf{p}(i) + \mathbf{n}_m(i) \tag{5},$$

but only, of course, if the landmark is within the visible range. Otherwise the measurement is just not available. We assume that the measurement noise $\mathbf{n}_m(i)$ is white noise, and uncorrelated with a standard deviation of $\sigma_n = 5$ m in each direction.

Each landmark is associated with a unique identification number. The measurements are provided by a Matlab cell array `Z`. The measurements of visible landmarks at time i are recorded in `Z{i}`. This cell is a structure with the following fields:

`Z{i}.id`	`A 1-dim array containing the identification numbers of landmarks that are visible at time i. The length of this array, M, is the number of visible landmarks. Note that M depends on i.`
`Z{i}.zpos`	`A 2xM array containing the measured x and y positions of the M visible landmarks. The measurement function is given in eq (5). The order of the landmarks is the same as in the identification array.`
`Z{i}.zbearing`	`A 1-dim array containing the bearings (in degrees) of the M landmarks (to be used in part IV).`

SLAM operates by embedding the positions of all the landmarks that are seen so far in the state vector. If a new landmark is detected, one which was never seen before, the state vector should be expanded. All corresponding matrices should be adapted as such. The measurement vector only embeds the measurements that are currently available. Therefore, the size of the vector may change during each cycle. The corresponding matrices should be adapted accordingly.

Questions:

4. Suppose that at a particular time i only two landmarks are visible. Suppose that the ids of the two landmarks are 200 and 21 (just an example). Suppose that both landmarks have been seen previously, and that their estimated position is embedded in the state vector at element 3 and 4 (landmark with id=21), and

7 and 8 (landmark with id 200). How do you define the measurement vector $\mathbf{z}(i)$ for this particular case? How is the corresponding measurement matrix $\mathbf{H}(i)$ defined? How is the covariance matrix of the measurement noise $\mathbf{C}_\mathbf{n}$ defined? How should the matrix $\mathbf{G}(\mathbf{u})$ be defined so that the process noise is properly modelled?

5. Suppose that at another instance of time, j, only one landmark is visible. The landmark has never been seen before. So, it must be a new one. The id assigned to that landmark is id=156. To be able to update the system with these measurements, the state vector must be augmented, and a prediction including this new landmark must be provided; the corresponding prediction covariance matrix must be adapted, and the system function needs to be redefined. Describe how (mathematical equation) these actions must be accomplished. **Note 1:** the augmentation of the state space is done just before the measurements are processed in the update step. Therefore, in principle you should not use the measurements during augmentation as these measurements will be used twice then. **Note 2** beware: the augmentation of the covariance matrix is tricky.
6. Create pseudocode that shows the main actions that you need to do in each cycle of the SLAM (prediction, updating, embedding of new landmarks, creation of a measurement vector and associated matrices, bookkeeping of the landmarks).
7. Implement the SLAM algorithm. Show the time development of the results of the algorithm in a graphical representation which is expanded and updated in each cycle of the algorithm. That is: in each cycle you add a marker in the plot that indicates the current estimated position of the track; you add new landmarks in the plot by means of its uncertainty region; you update the uncertainty regions of already existing landmarks. Now and then, you plot uncertainty regions of the track (as was done) in part I.
8. Make a mp4-movie of this visualization. See the Matlab functions `getframe` and `VideoWriter`. Also create a graph that you include in your report.
9. Compare the results with respect to the track to the 'prediction only' result.

Part III: smoothing

The Kalman filter is an online filter. In principle, each estimate is only based on past and present measurements, but not on future measurements. In offline processing, each estimate is therefore eligible for improvement since the 'future' measurements are not used. Retrodiction (smoothing) could be applied to improve the online filtered results. The most well-known smoother for fixed interval smoother is the Rauch-Tung-Striebel algorithm.

10. Implement the Rauch-Tung-Striebel algorithm. Visualize the results (movie+graph), and compare with the previous results.

Hints:

- The algorithm needs access to $\bar{\mathbf{x}}(i\,|\,i)$, $\mathbf{C}(i\,|\,i)$, $\bar{\mathbf{x}}(i+1\,|\,i)$, and $\mathbf{C}(i+1\,|\,i)$. You need to adapt you code of part II such that these variables are available afterwards. Since these vectors and matrices grow as time proceeds, their sizes depend on i. Therefore, it is not straightforward to store these variables in the usual arrays. Instead you might want to use `cellarrays`. These arrays allow storage with variable sizes.
- The Rauch-Tung-Striebel algorithm is as follows:

step 1: apply the (forward) Kalman filter from $i=0$ to $i=N$, yielding $\hat{\mathbf{x}}(i\,|\,i)$ and $\mathbf{C}(i\,|\,i)$
step 2: starting at $j=N-1$, apply a backward filter

(6)

RTS backward algorithm for one cycle from $j+1$ to j:

2.1 predicted state: $\hat{\mathbf{x}}(j+1|\,j)=\mathbf{F}\hat{\mathbf{x}}(j\,|\,j)$

2.2 covariance prediction: $\mathbf{C}(j+1|\,j)=\mathbf{F}\mathbf{C}(j\,|\,j)\mathbf{F}(i)^T+\mathbf{C}_\mathbf{w}(j)$

2.3 smoother gain: $\mathbf{B}=\mathbf{C}(j\,|\,j)\mathbf{F}(i)^T\mathbf{C}^{-1}(j+1|\,j)$

2.4 smoothed state: $\hat{\mathbf{x}}(j|N)=\hat{\mathbf{x}}(j\,|\,j)+\mathbf{B}\left[\hat{\mathbf{x}}(j+1|N)-\hat{\mathbf{x}}(j+1|\,j)\right]$

2.5 smooth covariance: $\mathbf{C}(j|N)=\mathbf{C}(j\,|\,j)+\mathbf{B}\left[\mathbf{C}(j+1|N)-\mathbf{C}(j+1|\,j)\right]\mathbf{B}^T$

Note that step 2.1 and 2.2 were already performed in the forward Kalman filter. If you have stored $\hat{\mathbf{x}}(i+1\,|\,i)$ and $\mathbf{C}(i+1\,|\,i)$ for each i, then you can reuse these, and step 2.1 and 2.2 are not needed.

- A direct implementation of the smoothing algorithm above, in the current situation, is difficult since the SLAM algorithm redefines regularly the state vector: each time a new landmark is discovered it expands the state vector. Suppose that:

$$\mathbf{x}(j)\in\mathbb{R}^P \text{ and } \mathbf{x}(j+1)\in\mathbb{R}^Q \tag{7}$$

and that $Q > P$ because at time $j+1$ new landmarks were discovered. Then:

- Define $\mathbf{F}(j)$ as a $P \times P$ matrix. That is, it only predicts the vehicle's state and the landmarks that were already known at time j, but not the newly discovered ones at time $j+1$.
- Use only the $P \times P$ submatrix of $\mathbf{C}(j+1|\,j)$. That is, use only the (co)variances of the old landmarks at time $j+1$, and not the (co)variances associated with the landmarks that were newly discovered at time $j+1$. By doing so, the matrix $\mathbf{B}$ in step 2.3 becomes $P \times P$.
- In step 2.4, use only the first P elements in the smoothed state vector $\hat{\mathbf{x}}(j+1|\,N)$ and predicted state $\hat{\mathbf{x}}(j+1|\,j)$.
- In step 2.5, use only the $P \times P$ submatrix of $\mathbf{C}(j+1|\,N)$. That is, only the (co)variances of the old landmarks at time $j+1$, and not the (co)variances associated with the newly discovered landmarks.

Part IV: bearing-only measurements[§§]

In this part, we assume that only the relative bearing of the landmarks are measured. For the m-th landmark with position $\mathbf{x}_m(i) = \begin{bmatrix} x_m(i) & y_m(i) \end{bmatrix}^T$, the relative bearing is the angle $\theta(i)$ that satisfies:

$$\begin{aligned} \cos\big(\theta(i)\big) &= \frac{x_m(i) - x(i)}{\sqrt{\big(x_m(i) - x(i)\big)^2 + \big(y_m(i) - y(i)\big)^2}} \\ \sin\big(\theta(i)\big) &= \frac{y_m(i) - y(i)}{\sqrt{\big(x_m(i) - x(i)\big)^2 + \big(y_m(i) - y(i)\big)^2}} \end{aligned} \tag{8}$$

i.e. the bearing is the 4-quadrant *arctan* function of the displacement between $\mathbf{x}_m(i)$ and $\mathbf{p}(i)$. In Matlab:

```
theta = atan2(ym-y,xm-x);
```
(9)

The bearing is given in the field `Z{i}.zbearing`, and is expressed in degrees. The uncertainty of the measured bearing is $\sigma_b = 2^0$.

Questions:

11. The measurement function, as expressed in eq (8) and (9), is nonlinear. We need extended Kalman filtering to solve the problem. An online linearization of the measurement function is needed. That is:

$$\begin{aligned} \mathbf{z}(i) &= \mathbf{h}\big(\mathbf{x}(i)\big) + \mathbf{n}(i) \\ &\approx \mathbf{h}\big(\hat{\mathbf{x}}(i)\big) + \mathbf{H}\big(\hat{\mathbf{x}}(i)\big)\big(\mathbf{x}(i) - \hat{\mathbf{x}}(i)\big) + \mathbf{n}(i) \end{aligned} \tag{10}$$

    $\mathbf{H}(\mathbf{x})$ is the Jacobian matrix of the measurement function $\mathbf{h}(\mathbf{x})$. Suppose that at a particular time i only two landmarks are visible. Suppose that the ids of the two landmarks are 200 and 21 (just an example). Suppose that both landmarks have been seen previously, and that their estimated position is embedded in the state vector at element 3 and 4 (landmark with id=21), and 7 and 8 (landmark with id 200). How do you define the measurement vector $\mathbf{z}(i)$ for this particular case? How is the corresponding measurement function $\mathbf{h}(\mathbf{x}(i))$ defined? How is the corresponding Jacobian matrix $\mathbf{H}(\mathbf{x}(i))$ defined? How is the covariance matrix of the measurement noise $\mathbf{C}_\mathbf{n}$ defined?
12. Implement the SLAM algorithm. Show the development of the algorithm in a graphical representation which is expanded and updated in each cycle of the algorithm. That is in each cycle: you add a marker in the plot that indicates the current estimated position of the track; you add new landmarks in the plot by means of its uncertainty region; you update the uncertainty regions of already existing landmarks. Now and then, you plot uncertainty regions of the track (as was done) in part I.
13. Make a movie of this visualization.
14. Compare the results with respect to the track to the previous results.

Hint:

- If a new landmark is discovered, the distance to that landmark is about 22 m. You may use that knowledge to initiate your estimate of the position of a new landmark.

[§§] This part is tough, and only meant as the ultimate challenge.