ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

151-0566-00 **Recursive Estimation** (Spring 2015)

Programming Exercise #1 Topic: Kalman Filtering

Issued: April 29, 2015 Due: May 27, 2015

Michael Muehlebach (michaemu@ethz.ch) , May 7, 2015

Extended Kalman Filter for Tracking a Three-Wheeled Robot

An Extended Kalman Filter (EKF) is to be designed for tracking the position and orientation of a three-wheeled robot that is moving on a plane. A schematic drawing of the robot is shown in Figure 1.

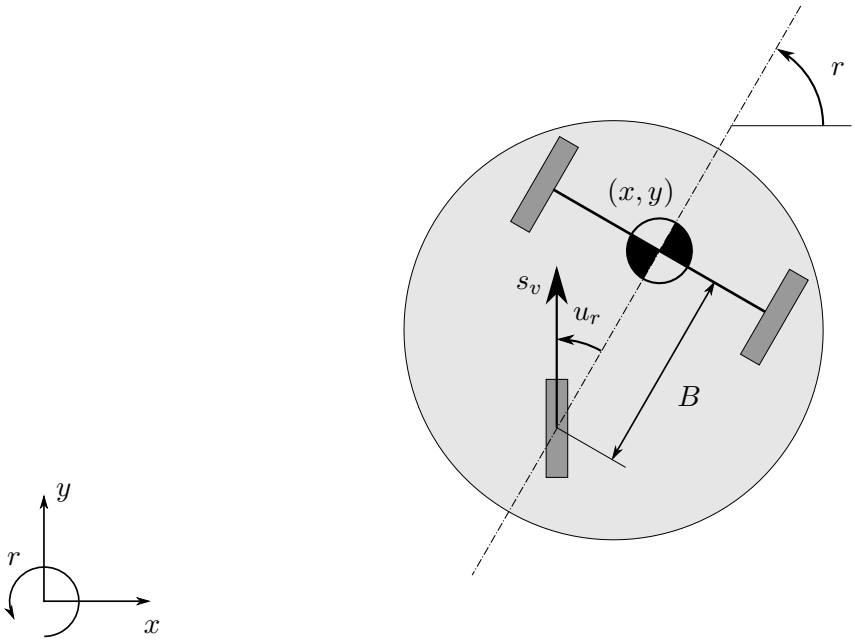

Figure 1: Top view of the three-wheeled robot and relevant physical quantities: (x, y) is the position of the robot's center of gravity, r its orientation, s_v denotes the (linear) velocity of the driving wheel, and u_r is the angle of the driving wheel.

The robot can command the angular velocity of its driving wheel ($u_v(t)$; in rad/s) and the driving wheel angle ($u_r(t)$; in rad). The translational speed s_t (in m/s) of the vehicle is

$$s_t(t) = s_v(t)\cos(u_r(t)) \tag{1}$$

with $s_v(t)$ being the driving wheel (linear) velocity,

$$s_v(t) = W\, u_v(t) \tag{2}$$

where W is the radius of the driving wheel (in m). The wheel radius is not known perfectly; it is modeled as a continuous random variable according to

$$W = W_0 + \gamma$$

with the known nominal wheel radius W_0 and the uniformly distributed random variable $\gamma \in [-\overline{\gamma}, \overline{\gamma}]$.

The rotational speed $s_r(t)$ (in rad/s) of the vehicle is

$$s_r(t) = \frac{-s_v(t)\sin(u_r(t))}{B} \tag{3}$$

where B is the wheel base (distance in m between the main axle and the driving wheel), see Figure 1.

The kinematic equations of the robot can be written as follows:

$$\dot{x}(t) = s_t(t)\cos\big(r(t)\big) \tag{4}$$
$$\dot{y}(t) = s_t(t)\sin\big(r(t)\big) \tag{5}$$
$$\dot{r}(t) = s_r(t) \tag{6}$$

where $(x(t), y(t))$ is the position of the robot (in m) and $r(t)$ its orientation (in rad).

The robot's commands are given at discrete time instants $t_0 = 0, t_1 = T, t_2 = 2T, \ldots$, where T is the constant sampling time in seconds. The robot's commands are kept constant over the sampling interval; that is, for example, $u_r(t) = u_r[k]$ for $t \in [kT, (k+1)T)$.

The robot starts at $(x(0), y(0)) = (x_0, y_0)$ with orientation $r(0) = r_0$. The initial position is uniformly distributed with $x_0, y_0 \in [-\bar{p}, \bar{p}]$, and the probability density function of r_0 uniformly distributed with $r_0 \in [-\bar{r}, \bar{r}]$.

At any sampling instant kT, the robot may receive noisy measurements of its orientation and its distance from the origin, i.e.

$$z_r[k] = r[k] + w_r[k]$$
$$z_d[k] = \sqrt{x^2[k] + y^2[k]} + w_d[k]$$

with $w_r[k] \sim \mathcal{N}\left(0, \sigma_r^2\right)$ and $w_d[k] \sim \mathrm{Tri}(\bar{w}_d)$, where $a \sim \mathcal{N}(0, \sigma^2)$ denotes a normally distributed continuous random variable with zero mean and variance σ^2, and $a \sim \mathrm{Tri}(\bar{a})$ denotes a continuous random variable with a triangular probability density function (see Figure 2). At any sampling instant kT, measurements may be available from one, two, or none of the sensors.

All random variables γ, r_0, x_0, y_0, $\{w_r[\cdot]\}$, $\{w_d[\cdot]\}$ are mutually independent.

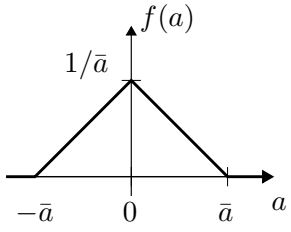

Figure 2: Triangular probability density function, $a \sim Tri(\bar{a})$.

Estimator Design Part 1

The objective is to design and tune a hybrid EKF to estimate the position and orientation of the three-wheeled robot. The estimator will be executed at the time instants $t_0 = 0, t_1 = T, t_2 = 2T, \ldots$. At time $t_k = kT$, the estimator has access to the time t_k, the control inputs $u_v[k-1]$ and $u_r[k-1]$, and possibly the measurements $z_d[k]$ and/or $z_r[k]$. Furthermore, the constants W_0, B, $\bar{\gamma}$, $\bar{w}_d$, σ_r, $\bar{p}$, and $\bar{r}$ are known to the estimator. The orientation, the position, and the wheel radius are estimator states. The motion of the robot is corrupted by process noise, the properties of which are unknown.

Design a hybrid EKF using the given problem data, and tune its performance by changing the process noise characteristics appropriately. We will evaluate your solution by executing it for several runs using the parameters given in the provided MATLAB files. Over all runs, the average root mean square error in the x and y position trajectory will be computed. You will receive full marks on this part if the average error is below 0.6m.

Estimator Design Part 2

The objective is to design a hybrid EKF for a different robot of similar type. For this second robot, a model of the process noise is available. This model takes non-idealities in the actuation mechanism (for example, wheel slip or imperfect command tracking) into account. That is, Equation (2) is *replaced* by

$$s_v(t) = W\, u_v(t)\,\big(1 + v_v(t)\big) \tag{7}$$

where $v_v(t)$ is assumed to be continuous-time white noise with

$$E[v_v(t)] = 0, \qquad E[v_v(t)v_v(t+\tau)] = Q_v\delta(\tau),$$

for all times. The Dirac pulse (see lecture 9 on the EKF) is denoted by $\delta(\tau)$. Equations (1) and (3) are *replaced* by

$$s_t(t) = s_v(t)\cos(u_r(t) + v_r(t)) \quad \text{and} \quad s_r(t) = \frac{-s_v(t)\sin(u_r(t) + v_r(t))}{B}\,. \tag{8}$$

with $v_r(t)$ assumed to be continuous-time white noise with

$$E[v_r(t)] = 0, \qquad E[v_r(t)v_r(t+\tau)] = Q_r\delta(\tau),$$

for all times. The remainder of the system description remains unchanged. The process noise signals $\{v_v(\cdot)\}$ and $\{v_r(\cdot)\}$ are mutually independent, and independent of all other random variables. Additionally to the information of the EKF in design part 1 above, this EKF has access to the constants Q_v and Q_r.

Design a hybrid EKF that explicitly takes into account the above process noise model. For evaluating your solution, we will test it on the given problem data. Moreover, we will make suitable modifications to all parameters.

Provided Matlab Files

A set of Matlab files is provided on the class website. Please use them for solving the exercise.

`run.m`	Matlab function that is used to execute a simulation of the true system, run the estimator, and display the results. The single input argument `designPart` is used to specify the estimator design part (1 or 2).
`Estimator.m`	Matlab function template to be used for your implementations of the EKFs. Use this file for both estimator design parts. You can use the input argument `designPart` to distinguish between the two.
`Simulator.p`	Matlab function used to simulate the motion of the robot and measurements. This function is called by `run.m`, and is obfuscated (i.e., its source code is not readable).
`KnownConstants.m`	Constants known to the estimator. For the evaluation of the design part 2, these may be modified.
`UnknownConstants.m`	Constants *not* known to the estimator. For the evaluation of the design part 2, these may be modified.

Task

Implement your solutions for the estimator design part 1 and part 2 in the file `Estimator.m`. Your code has to run with the Matlab function `run.m` (for both parts, that is, `run(1)` and `run(2)`). Use exactly the function definition as given in the template `Estimator.m` for the implementation of your estimators.

To judge your own solution, a typical performance of an EKF implementation for part 1 and the problem data provided in `KnownConstants.m` and `UnknownConstants.m` is shown in Fig. 3 and 4. To reproduce the exact data used for these plots, the seed of the random number generator can be fixed in the file `run.m` (lines 62-63). Note that the results shown in the plots do not represent the best possible performance.

Deliverables

Note: Your submission has to follow these instructions exactly, as grading is automated. Submissions that do not conform will have points deducted.

Up to three students are allowed to work together on the programming exercise. They will all receive the same grade.

As a group, you must read and understand the ETH plagiarism policy here:
`http://www.plagiarism.ethz.ch/` – each submitted work will be tested for plagiarism. You must fill out the *Declaration of Orignality*, available here:
`http://tiny.cc/ETHPlagiarismForm`.

Hand in a single zip-file, where the filename contains the names of all team-members according to this template (note that there are no spaces in the filename):
`RE15Ex1_Firstname1Surname1_Firstname2Surname2_Firstname3Surname3.zip`.

The zip-file contains only two files:

1. Your implementation of the estimator design part 1 and part 2 in the single file `Estimator.m`, which has the exact same function definition as in the provided template.

2. A scan of the *Declaration of Originality*, signed by all team-members.

Send your zip-file in a single email:

- Use this *exact* subject: `programming exercise 1 submission`
- In the email body, include a list of all team-member names.
- Send the email to `re2015@ethz.ch`.

You will receive an automatic reply confirming receipt of the email. A human will not look at the email before the deadline expires, and you are ultimately responsible that we correctly receive your solution in time. Late submissions will not be considered, nor will submissions where the attachment was forgotten.

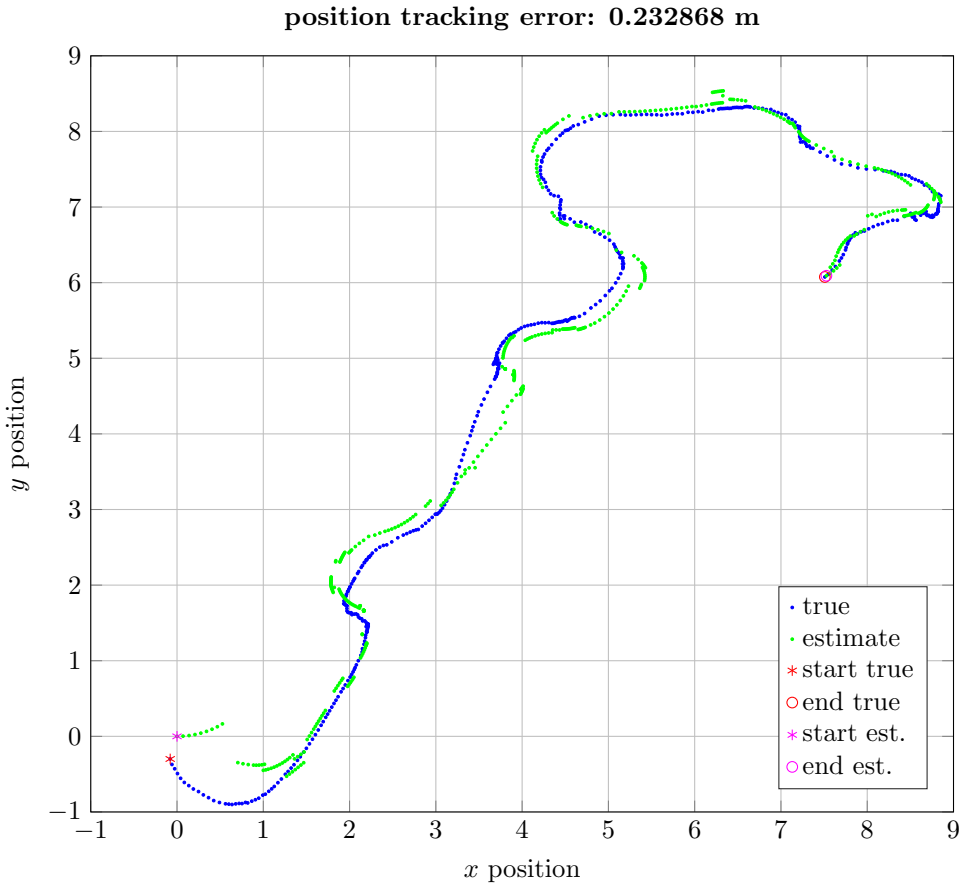

Figure 3: Typical tracking performance of an EKF implementation for design part 1 and the provided problem data (the seeds of the the random number generators were initialized to fixed values, as can be done in `run.m` *by uncommenting the corresponding lines). The shown position tracking error is the root mean square of the error in x and y position trajectory.*

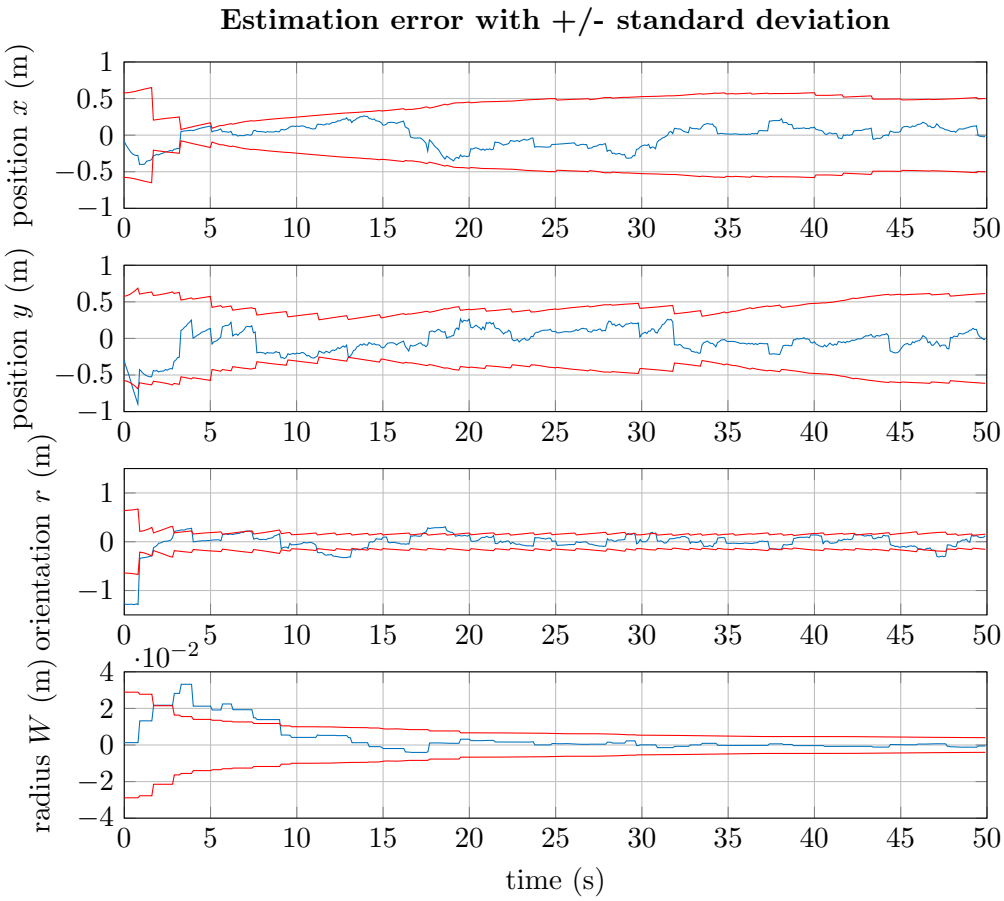

Figure 4: Estimation errors (blue) with +/- one standard deviation (red) for the estimator data shown in Figure 3.