ME8135 — Assignment 2 Solution

Student: Arash Basirat Tabrizi
Submitted to: Dr. Sajad Saeedi
Due: June 2, 2023

---

1. Problem Statement:

We wish to use PyGame to simulate a simplified 2D robot and perform state estimation using a Kalman Filter. The robot's Motion Model is defined as follows:

$$\dot{x} = \frac{r}{2}(u_r + u_l) + w_x, \dot{y} = \frac{r}{2}(u_r + u_l) + w_y \tag{1}$$

In equation 1, r = 0.1 m, is the radius of the wheel, u_r and u_l are control signals applied to the right and left wheels. Additionally, $w_x = N(0, 0.1)$ and $w_y = N(0, 0.15)$.

We wish to simulate the system such that the robot is driven 1 m to the right. We are to assume the speed of each wheel is fixed at $0.1\frac{m}{s}$. We use the following initial values:

$$x_0 = 0, y_0 = 0, P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \text{(initial covariance matrix)} \tag{2}$$

Also, we are to assume that the motion model is computed 8 times a second. We assume every second a measurement is given:

$$z = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} r_x \\ r_y \end{bmatrix} \tag{3}$$

Where $r_x = N(0, 0.05)$ and $r_y = N(0, 0.075)$.

We wish to plot and animate the trajectory along with covariance ellipse for all motion models and measurement updates. We also aim to report what happens when a measurement arrives. We should note to discretize the system: $\dot{x} = \frac{(x_k - x_{k-1})}{T}$ where T is 1/8 sec. In the animation, we will show both ground-truth and estimates.

Solution Formulation: From the discretized system model we have:

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{r_x}{2} \\ \frac{r_y}{2} \end{bmatrix} \tag{4}$$

where $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = A$ is our *state transition matrix*. Since the discretized system is formulated as follows:

$$\begin{aligned} \frac{x_k - x_{k-1}}{T} &= \frac{r}{2}(u_{r,k} + u_{l,k}) + w_{x,k} \\ x_k &= x_{k-1} + \frac{r}{2}(u_{r,k} + u_{l,k})T + w_{x,k}T \\ \frac{y_k - y_{k-1}}{T} &= \frac{r}{2}(u_{r,k} + u_{l,k}) + w_{y,k} \\ y_k &= y_{k-1} + \frac{r}{2}(u_{r,k} + u_{l,k})T + w_{y,k}T \end{aligned} \tag{5}$$

Comparing this with the general state transition equation $\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_k + \mathbf{w}_k$, we identify the *control*

matrix, B, to be:

$$B = \begin{bmatrix} \frac{r}{2}T & \frac{r}{2}T \\ \frac{r}{2}T & \frac{r}{2}T \end{bmatrix} \tag{6}$$

Note that in the state transition equation, $\mathbf{u}_k = \begin{bmatrix} u_{r,k} \\ u_{l,k} \end{bmatrix}$ is the control vector and $\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$ is the state vector. $\mathbf{w}_k$ is a Gaussian random vector that models the uncertainty introduced by the state transition.

To determine the *process noise covariance matrix*, Q, of the system, we need to consider the variances of the individual process noise components:

$$Q = \begin{bmatrix} w_x & 0 \\ 0 & w_y \end{bmatrix} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.15 \end{bmatrix} \tag{7}$$

Similarly, from the general measurement equation $\mathbf{z} = C\mathbf{x} + \mathbf{n}$, the *measurement noise covariance matrix*, R, of the system is as follows:

$$R = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.075 \end{bmatrix} \tag{8}$$

Additionally, we deduce the *observation matrix*, C, to be:

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \tag{9}$$

Thus, the state transition model in matrix form is as follows:

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{r}{2}T & \frac{r}{2}T \\ \frac{r}{2}T & \frac{r}{2}T \end{bmatrix} \begin{bmatrix} u_{r,k} \\ u_{l,k} \end{bmatrix} + \begin{bmatrix} w_{x,k} \\ w_{y,k} \end{bmatrix} T \tag{10}$$

We will use the Kalman Filter algorithm presented in chapter 3 of the Probabilistic Robotics textbook, Figure 1, for the implementation of our solution. The algorithm is explicitly referenced in sections of our Python simulation code.

Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**

$\bar{\mu}_t = A_t\, \mu_{t-1} + B_t\, u_t$

$\bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^T + R_t$

$K_t = \bar{\Sigma}_t\, C_t^T (C_t\, \bar{\Sigma}_t\, C_t^T + Q_t)^{-1}$

$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\, \bar{\mu}_t)$

$\Sigma_t = (I - K_t\, C_t)\, \bar{\Sigma}_t$

return μ_t, Σ_t

Figure 1: The KF algorithm for linear Gaussian state transitions and measurements.

The algorithm represents the belief $bel(x_t)$ at time t by the mean μ_t and the covariance Σ_t (output of the KF). The input of the KF is the belief at time $t-1$, represented by μ_{t-1} and Σ_{t-1}. The predicted belief at time t is represented by $\bar{\mu}_t$ and $\bar{\Sigma}_t$.

Simulation Results and Observations:

The simulation animation can be found at: *GitHub Repo*/A2/animation_1.gif

The animation has been slowed down for demonstration purposes.

Due to the uncertainty, $\mathbf{w}_k$, introduced by the state transition model, noise is accounted for in both the x and y dimensions. The model suggests that there is 50% higher variance for the random noise in the y direction. As a result of which the robot drifts more along the y direction in our simulation. The same observation can be made about the noise in the measurement model, however, measurement noise variance is smaller than of the input noise variance. The smaller uncertainty in the measurements is a valid and accurate representation of real-life robots.

The trajectory plots generated by our Python simulation code (A2/Q1.py) are illustrated as follows:

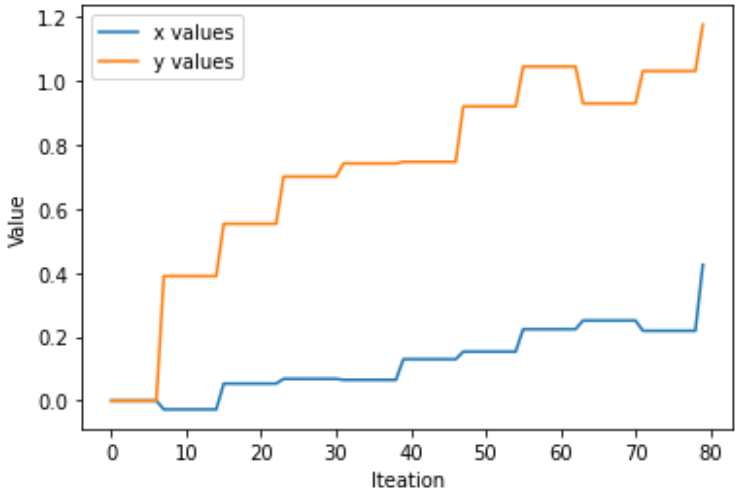

Figure 2: Measurement trajectories during simulation.

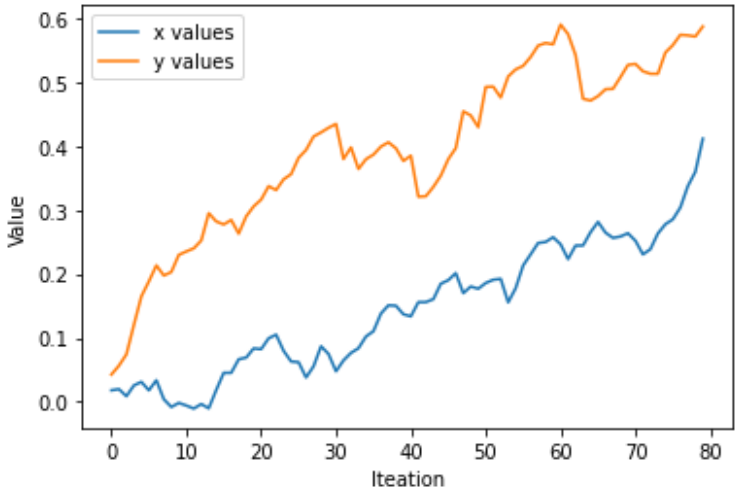

Figure 3: Predicted trajectories during simulation.

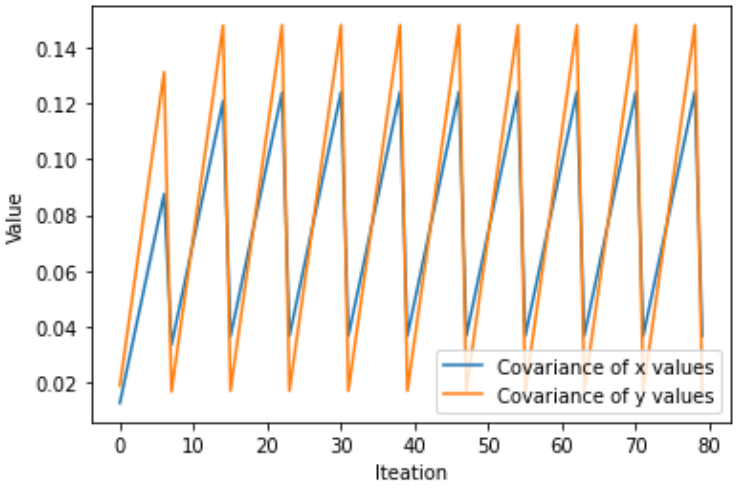

Figure 4: Covariance trajectories during simulation.

We observe that 8 predictions and 1 measurement are made per iteration. The predicted position drifts with increasingly larger covariance, as illustrated by the covariance ellipse animation, before a measurement is made to correct the robot's position. The trajectory calculated by the measurement steps (red trajectory in animation) appears to be more linear than the motion model's ground-truth trajectory (green trajectory in animation).

2. Problem Statement:

We wish to repeat the previous question, this time with a classic motion model and range observations made from a landmark located at M = [10,10]. L is the distance between the wheel, known as wheelbase, and is 0.3m. We must program the robot such that it loops around point M.

In *part(a)*, the classic motion model presented in equation 13 is used in combination with the linear measurement model previously presented in equation 3. In *part(b)*, the same classic motion model is used in combination with the range/bearing measurements of point M. Where we assume range noise is N(0,0.1) and bearing noise is N(0,0.01). Range is in meters, and bearing is in radians.

$$\dot{x} = \frac{r}{2}\left(u_r + u_l\right) cos(\theta) + w_x, \dot{y} = \frac{r}{2}\left(u_r + u_l\right) sin(\theta) + w_y, \dot{\theta} = \frac{r}{L}\left(u_r - u_l\right) \tag{11}$$

$$u_w = \frac{1}{2}\left(u_r + u_l\right), u_\psi = \left(u_r - u_l\right) \tag{12}$$

Combining equations 11 and 12 gives:

$$\dot{x} = r u_w cos(\theta) + w_w, \dot{y} = r u_w sin(\theta) + w_w, \dot{\theta} = \frac{r}{L} u_\psi + w_\psi \tag{13}$$

Where $n_\psi = N(0, 0.01)$ and $n_d = N(0, 0.1)$.

Part (a) Solution Formulation:

The general state transition equation is now $\mathbf{x}_k = \text{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)$. The function f(.) represents the nonlinear motion model.

For the robot to rotate, angle (θ) is integrated into the speed equations with bearing noise. To model the motion properly we discretize as follows:

$$\begin{aligned}\dot{\theta}_k &= \frac{r}{L}u_\psi + w_\psi \\ &= \frac{\theta_k - \theta_{k-1}}{T} \\ \theta_k &= T\dot{\theta}_k + \theta_{k-1}\end{aligned} \tag{14}$$

$$\begin{aligned}\dot{\mathrm{x}}_k &= \begin{bmatrix} ru_w cos(\theta) + w_w \\ ru_w sin(\theta) + w_w \end{bmatrix} \\ &= \frac{\mathrm{x}_k - \mathrm{x}_{k-1}}{T} \\ \mathrm{x}_k &= T\dot{\mathrm{x}}_k + \mathrm{x}_{k-1} \\ &= \mathrm{f}(\mathrm{x}_{k-1}, \mathrm{v}_k, \mathrm{w}_k)\end{aligned} \tag{15}$$

In equation 15, $\mathrm{v}_k = T\dot{\mathrm{x}}_{k-1}$ and $\mathrm{w}_k = \begin{bmatrix} w_w \\ w_\psi \end{bmatrix}$.

EKF: The Extended Kalman Filter (EKF) calculates a Gaussian approximation to the true belief. Thus, the EKF inherits from the Kalman Filter the basic belief representation, but it differs in that this belief is only approximate, not exact as was the case in KFs. The goal of the EKF is thus shifted from computing the exact posterior to efficiently estimating its mean and covariance. However, since these statistics cannot be computed in closed form, the EKF has to resort to an additional approximation. This additional approximation is the concept of *linearization*.

For our nonlinear motion model, linearization approximates the nonlinear function f(.) by a linear function that is tangent to f(.) at the mean of a Gaussian. We will use the textbook's techniques for linearizing our nonlinear motion model as described in chapter 4 of *State Estimation for Robotics* (equations 4.24a, 4.25a, 4.25b in the textbook):

$$\mathrm{f}(\mathrm{x}_{k-1}, \mathrm{v}_k, \mathrm{w}_k) \approx \breve{\mathrm{x}}_k + \mathrm{F}_{k-1}(\mathrm{x}_{k-1} - \breve{\mathrm{x}}_{k-1}) + \acute{\mathrm{w}}_k \tag{16}$$

$$\breve{\mathrm{x}}_k = \mathrm{f}(\mathrm{x}_{k-1}, \mathrm{v}_k, 0) = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{y}_{k-1} \end{bmatrix} + T \begin{bmatrix} ru_w cos(T\frac{r}{L}u_\psi + \theta_{k-1}) \\ ru_w sin(T\frac{r}{L}u_\psi + \theta_{k-1}) \end{bmatrix} \tag{17}$$

$$\mathrm{F}_{k-1} = \left.\frac{\partial \mathrm{f}(\mathrm{x}_{k-1}, \mathrm{v}_k, \mathrm{w}_k)}{\partial \mathrm{x}_{k-1}}\right|_{\hat{\mathrm{x}}_{k-1}, \mathrm{v}_k, 0} = \left.\frac{\partial (T\dot{\mathrm{x}}_k + \mathrm{x}_{k-1})}{\partial \mathrm{x}_{k-1}}\right|_{\hat{\mathrm{x}}_{k-1}, \mathrm{v}_k, 0} = 1 \tag{18}$$

$$\acute{\mathrm{w}}_k = \left.\frac{\partial \mathrm{f}(\mathrm{x}_{k-1}, \mathrm{v}_k, \mathrm{w}_k)}{\partial \mathrm{w}_k}\right|_{\hat{\mathrm{x}}_{k-1}, \mathrm{v}_k, 0} \mathrm{w}_k = \begin{bmatrix} \frac{\partial \mathrm{f}_x}{\partial w_w} & \frac{\partial \mathrm{f}_x}{\partial w_\psi} \\ \frac{\partial \mathrm{f}_y}{\partial w_w} & \frac{\partial \mathrm{f}_y}{\partial w_\psi} \end{bmatrix} \begin{bmatrix} w_w \\ w_\psi \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathrm{f}_x}{\partial w_w} w_w + \frac{\partial \mathrm{f}_x}{\partial w_\psi} w_\psi \\ \frac{\partial \mathrm{f}_y}{\partial w_w} w_w + \frac{\partial \mathrm{f}_y}{\partial w_\psi} w_\psi \end{bmatrix} \tag{19}$$

$$\begin{aligned}\frac{\partial \mathrm{f}}{\partial w_w} &= T \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ \frac{\partial \mathrm{f}}{\partial w_\psi} &= \frac{\partial \mathrm{f}}{\partial w_\psi} = T^2 r u_w \begin{bmatrix} -sin(T\frac{r}{L}u_\psi + \theta_{k-1}) \\ cos(T\frac{r}{L}u_\psi + \theta_{k-1}) \end{bmatrix}\end{aligned} \tag{20}$$

We define $\acute{\mathbf{Q}}_k$ as follows using equation 19 ($\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$):

$$\acute{\mathbf{Q}}_k = \begin{bmatrix} Var(w_x) & Cov(w_x, w_y) \\ Cov(w_x, w_y) & Var(w_y) \end{bmatrix} = \begin{bmatrix} 0.1\frac{\partial \mathrm{f}_x}{\partial w_w} + 0.01\frac{\partial \mathrm{f}_x}{\partial w_\psi} & 0 \\ 0 & 0.1\frac{\partial \mathrm{f}_y}{\partial w_w} + 0.01\frac{\partial \mathrm{f}_y}{\partial w_\psi} \end{bmatrix} \tag{21}$$

Since our measurement model remains linear, we do not need to linearize it. We must note that for the measurement model, the observation matrix, C, is replaced with the Jacobian when implementing the EKF algorithm.

Part (b) Solution Formulation:

The general measurement equation is now $\mathbf{z}_k = \mathrm{g}(\mathbf{x}_k, \mathbf{n}_k)$. The function g(.) represents the nonlinear observation model.

For our range/bearing (d, ϕ) measurement model we have:

$$d(\mathrm{x}_k, n_d) = d(x_k, y_k, n_d) = \sqrt{(x_k - 10)^2 - (y_k - 10)^2} + n_d \tag{22}$$

$$\phi_k(\theta_k, n_\phi) = \frac{\pi}{2} + \theta_k + n_\phi \tag{23}$$

We formulate the measurement model as follows:

$$\mathrm{z}_k = \mathrm{g}(\mathrm{x}_k, \mathrm{n}_k) = \begin{bmatrix} 10 + dcos(\phi_k) \\ 10 + dsin(\phi_k) \end{bmatrix}, \mathrm{n}_k = \begin{bmatrix} n_d \\ n_\phi \end{bmatrix} \tag{24}$$

In equation 24, $n_d = N(0, 0.1)$ and $n_\phi = N(0, 0.01)$.

Following the procedure presented in chapter 4 of *State Estimation for Robotics* textbook, we linearize our nonlinear measurement model (equations 4.24b, 4.26a, 4.26b in the textbook):

$$\mathrm{g}(\mathrm{x}_k, \mathrm{n}_k) \approx \breve{\mathrm{z}}_k + \mathbf{G}_k(\mathrm{x}_k - \breve{\mathrm{x}}_k) + \acute{\mathbf{n}}_k \tag{25}$$

$$\breve{\mathrm{z}}_k = \mathrm{g}(\breve{\mathrm{x}}_k, 0) \tag{26}$$

$$\mathbf{G}_k = \left.\frac{\partial \mathrm{g}(\mathrm{x}_k, \mathbf{n}_k)}{\partial \mathrm{x}_k}\right|_{\breve{\mathrm{x}}_k, 0} = \frac{1}{\sqrt{(\breve{x}_k - 10)^2 - (\breve{y}_k - 10)^2}} \begin{bmatrix} (\breve{x}_k - 10)^2 cos(\phi_k) & (\breve{y}_k - 10)^2 cos(\phi_k) \\ (\breve{x}_k - 10)^2 sin(\phi_k) & (\breve{y}_k - 10)^2 sin(\phi_k) \end{bmatrix} \tag{27}$$

$$\acute{\mathbf{n}}_k = \left.\frac{\partial \mathrm{g}(\mathrm{x}_k, \mathbf{n}_k)}{\partial \mathrm{n}_k}\right|_{\breve{\mathrm{x}}_k, 0} \mathbf{n}_k = \left.\begin{bmatrix} n_d cos(\phi_k) - dn_\phi sin(\phi_k) \\ n_d sin(\phi_k) + dn_\phi sin(\phi_k) \end{bmatrix}\right|_{d(\breve{\mathrm{x}}_k, 0), \phi_k(\theta_k, 0)} \tag{28}$$

We define $\acute{\mathbf{R}}_k$ as follows using equation 28 ($\mathbf{n}_k \sim N(0, \mathbf{R}_k)$):

$$\acute{\mathbf{R}}_k = \begin{bmatrix} Var(n_x) & Cov(n_x, n_y) \\ Cov(n_x, n_y) & Var(n_y) \end{bmatrix} = \begin{bmatrix} 0.1cos(\phi_k) - 0.01dsin(\phi_k) & 0 \\ 0 & 0.1sin(\phi_k) + 0.01dsin(\phi_k) \end{bmatrix} \tag{29}$$

For part(b), both motion and observation models are nonlinear, hence, the EKF algorithm becomes:

$$
\begin{aligned}
\text{predictor:} \quad & \check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_k', && \text{(4.32a)} \\
& \check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{v}_k, \mathbf{0}), && \text{(4.32b)} \\
\text{Kalman gain:} \quad & \mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{G}_k^T\left(\mathbf{G}_k\check{\mathbf{P}}_k\mathbf{G}_k^T + \mathbf{R}_k'\right)^{-1}, && \text{(4.32c)} \\
\text{corrector:} \quad & \hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{G}_k)\,\check{\mathbf{P}}_k, && \text{(4.32d)} \\
& \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\underbrace{(\mathbf{z}_k - \mathbf{g}(\check{\mathbf{x}}_k, \mathbf{0}))}_{\text{innovation}}. && \text{(4.32e)}
\end{aligned}
$$

Figure 5: Classic recursive update equations for the EKF from *State Estimation for Robotics* textbook.

In Figure 5, there are Jacobians embedded in the $\acute{\mathbf{Q}}_k$ and $\acute{\mathbf{R}}_k$ covariances for the noise. This comes from the fact that we allowed the noise to be applied within the nonlinearities.

Simulation Results and Observations: