UNIVERSITÉ
BOURGOGNE
EUROPE

Assignment 1

Mariam JANBEIN , Adel KANSO

May 16, 2025

1 Kalman Filter Correction as Product of Gaussians

To show that the Kalman Filter correction step is equivalent to multiplying the predicted state and observation Gaussians, we compare their resulting mean and variance.

1.1 Predicted State and Observation Gaussians:

- Predicted state: $\mathcal{N}(x; \bar{\mu}, \bar{\sigma}^2)$
- Observation: $\mathcal{N}(x; z, \sigma_{\text{obs}}^2)$

1.2 Product of Gaussians:

The product of two Gaussians $\mathcal{N}(x; \mu_1, \sigma_1^2)$ and $\mathcal{N}(x; \mu_2, \sigma_2^2)$ is proportional to:

$$\mathcal{N}\left(x; \frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\mu_2, \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}}\right)$$

1.3 Applying to Our Case:

$$\mu_1 = \bar{\mu}, \quad \sigma_1^2 = \bar{\sigma}^2$$
$$\mu_2 = z, \quad \sigma_2^2 = \sigma_{\text{obs}}^2$$

1.4 Resulting Mean and Variance:

- Mean:
$$\frac{\sigma_{\text{obs}}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}\bar{\mu} + \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}z$$
- Variance:
$$\frac{\bar{\sigma}^2\sigma_{\text{obs}}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}$$

1.5 Kalman Filter Correction Equations:

- Kalman Gain:
$$K = \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}$$
- Updated Mean:
$$\mu = \bar{\mu} + K(z - \bar{\mu})$$

  Substituting K:
$$\mu = \bar{\mu} + \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}(z-\bar{\mu}) = \frac{\sigma_{\text{obs}}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}\bar{\mu} + \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}z$$
- Updated Variance:
$$\sigma^2 = (1-K)\bar{\sigma}^2$$

  Substituting K:
$$\sigma^2 = \left(1 - \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}\right)\bar{\sigma}^2 = \frac{\bar{\sigma}^2\sigma_{\text{obs}}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}$$

1.6 Conclusion:

The mean and variance from the Kalman Filter correction step match exactly those obtained by multiplying the predicted state and observation Gaussians. Thus, the Kalman Filter correction is equivalent to combining these Gaussians multiplicatively, up to a scaling factor.

$$\mu = \bar{\mu} + K(z-\bar{\mu}), \quad \text{where } K = \frac{\bar{\sigma}^2}{\bar{\sigma}^2+\sigma_{\text{obs}}^2}$$
$$\sigma^2 = (1-K)\bar{\sigma}^2$$

2 Jacobian Motion Model

To derive the Jacobians $G = \frac{\partial g}{\partial s}$ and $V = \frac{\partial g}{\partial u}$ for the given motion model, we start by analyzing the motion model equations:

$$\begin{aligned} x_{t+1} &= x_t + \delta_{trans}\cos(\theta_t + \delta_{rot1}) \\ y_{t+1} &= y_t + \delta_{trans}\sin(\theta_t + \delta_{rot1}) \\ \theta_{t+1} &= \theta_t + \delta_{rot1} + \delta_{rot2} \end{aligned}$$

2.1 Jacobian G (with respect to the state s):

Each entry $G[i][j]$ is the partial derivative of the i-th component of s_{t+1} with respect to the j-th component of s_t:

- For x_{t+1}:
  - $$\frac{\partial x_{t+1}}{\partial x_t} = 1$$
  - $$\frac{\partial x_{t+1}}{\partial y_t} = 0$$
  - $$\frac{\partial x_{t+1}}{\partial \theta_t} = -\delta_{trans}\sin(\theta_t + \delta_{rot1}]$$
- For y_{t+1}:
  - $$\frac{\partial y_{t+1}}{\partial x_t} = 0$$
  - $$\frac{\partial y_{t+1}}{\partial y_t} = 1$$
  - $$\frac{\partial y_{t+1}}{\partial \theta_t} = \delta_{trans}\cos(\theta_t + \delta_{rot1}]$$
- For θ_{t+1}:
  - $$\frac{\partial \theta_{t+1}}{\partial x_t} = 0$$
  - $$\frac{\partial \theta_{t+1}}{\partial y_t} = 0$$
  - $$\frac{\partial \theta_{t+1}}{\partial \theta_t} = 1$$

Thus, the Jacobian G is:

$$G = \begin{bmatrix} 1 & 0 & -\delta_{trans}\sin(\theta_t + \delta_{rot1}) \\ 0 & 1 & \delta_{trans}\cos(\theta_t + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

2.2 Jacobian V (with respect to the control u):

Each entry $V[i][j]$ is the partial derivative of the i-th component of s_{t+1} with respect to the j-th component of u:

- For x_{t+1}:
  - $$\frac{\partial x_{t+1}}{\partial \delta_{rot1}} = -\delta_{trans}\sin(\theta_t + \delta_{rot1})$$
  - $$\frac{\partial x_{t+1}}{\partial \delta_{trans}} = \cos(\theta_t + \delta_{rot1}]$$
  - $$\frac{\partial x_{t+1}}{\partial \delta_{rot2}} = 0$$
- For y_{t+1}:
  - $$\frac{\partial y_{t+1}}{\partial \delta_{rot1}} = \delta_{trans}\cos(\theta_t + \delta_{rot1})$$
  - $$\frac{\partial y_{t+1}}{\partial \delta_{trans}} = \sin(\theta_t + \delta_{rot1})$$
  - $$\frac{\partial y_{t+1}}{\partial \delta_{rot2}} = 0$$
- For
  $$\theta_{t+1}$$
  :
  - $$\frac{\partial \theta_{t+1}}{\partial \delta_{rot1}} = 1$$
  - $$\frac{\partial \theta_{t+1}}{\partial \delta_{trans}} = 0$$
  - $$\frac{\partial \theta_{t+1}}{\partial \delta_{rot2}} = 1$$

Thus, the Jacobian $\mathbf{V}$ is:

$$\mathbf{V} = \begin{bmatrix} -\delta_{trans}\sin(\theta_t + \delta_{rot1}) & \cos(\theta_t + \delta_{rot1}) & 0 \\ \delta_{trans}\cos(\theta_t + \delta_{rot1}) & \sin(\theta_t + \delta_{rot1}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

2.3 Final Answer

The Jacobian matrices are:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & -\delta_{trans}\sin(\theta_t + \delta_{rot1}) \\ 0 & 1 & \delta_{trans}\cos(\theta_t + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} -\delta_{trans}\sin(\theta_t + \delta_{rot1}) & \cos(\theta_t + \delta_{rot1}) & 0 \\ \delta_{trans}\cos(\theta_t + \delta_{rot1}) & \sin(\theta_t + \delta_{rot1}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

3 EKF & PF

3.1 ekf.py: Extended Kalman Filter Implementation

3.1.1 Class Definition

Defines an `ExtendedKalmanFilter` class.

3.1.2 Initialization

The constructor (`__init__`) initializes the EKF with:

- Mean state estimate
- Covariance matrix
- Noise parameters: `alphas` (motion model) and `beta` (observation model)

3.1.3 Reset Method

Reinitializes the filter's state and covariance.

3.1.4 Update Method

Performs prediction and correction steps.

1. Prediction Step
   - Computes Jacobians G and V for the motion model.
   - Predicts next state and covariance:

$$\bar{\mu} = \texttt{env.forward(self.mu.ravel(), u.ravel())} \tag{1}$$
$$\bar{\Sigma} = G\Sigma G^T + VRV^T \tag{2}$$

2. Correction Step
   - Computes expected observation $\hat{z}$ and Jacobian H.
   - Calculates innovation: actual minus predicted observation.
   - Updates state estimate and covariance:

$$K = \bar{\Sigma}H^T(H\bar{\Sigma}H^T + Q)^{-1} \tag{3}$$
$$\bar{\mu} + = K \cdot \texttt{innovation} \tag{4}$$
$$\bar{\Sigma} = (I - KH)\bar{\Sigma} \tag{5}$$

3.2 pf.py: Particle Filter Implementation

3.2.1 Class Definition

Defines a `ParticleFilter` class.

3.2.2 Initialization

The constructor (`__init__`) initializes the PF with:

- Mean state estimate
- Covariance matrix
- Number of particles
- Noise parameters (`alphas`, `beta`)

3.2.3 Reset Method

Initializes particles using a Gaussian distribution around the initial mean and covariance.

3.2.4 Update Method

Performs prediction and correction:

1. Prediction Step Each particle is propagated with noise:
   ```
   self.particles[i, :] = env.forward(self.particles[i], env.sample_noisy_action(u, se
   ```
2. Weight Calculation Based on the likelihood of predicted observation:
   ```
   self.weights[i] = env.likelihood(innovation, self.beta)
   ```
3. Resampling Low-variance resampling algorithm is applied to eliminate low-weight particles and duplicate high-weight ones.
4. State Estimation Estimates mean and covariance of the particles:
   ```
   mean, cov = self.mean_and_variance(self.particles)
   ```

3.3 Summary

EKF: Uses a Gaussian state distribution. Linearizes motion and observation models using Jacobians. Efficient but assumes linearity and Gaussian noise.

PF: Represents state with particles. Handles non-linear and non-Gaussian noise. More flexible but computationally intensive.

4 Analysis

4.1 EKF Analysis (analyze_ekf.py)

4.1.1 Mean Position Error vs Noise Factor (Part b)

This plot illustrates the mean position error of the Extended Kalman Filter (EKF) under varying noise factors. The noise factor r scales the motion and observation noise in the environment.

- A lower mean position error indicates higher localization accuracy.
- As the noise factor increases, the mean position error generally **decreases**.
- Excessive noise can cause the mean error to **increase** again, indicating degraded performance.

4.1.2 ANEES vs Noise Factor (Part c)

ANEES (Average Normalized Estimation Error Squared) assesses the consistency between the filter's covariance matrix and the actual estimation error.

- ANEES ≈ 1 indicates good consistency.
- As the noise factor increases, ANEES initially **decreases** and then stabilizes around 0.
- Very high noise factors may lead to underestimated covariance, causing ANEES to drop too low.

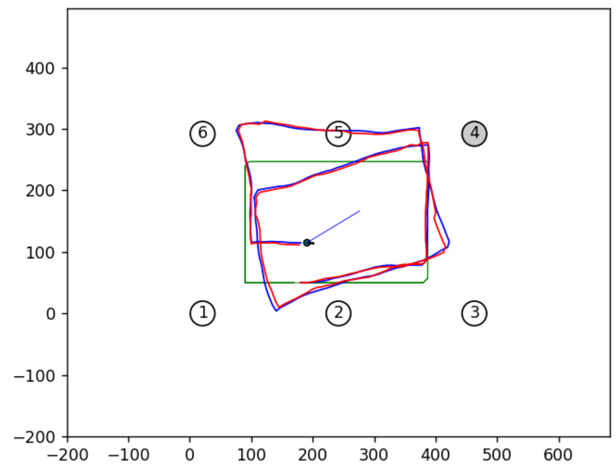

Figure 1: Path

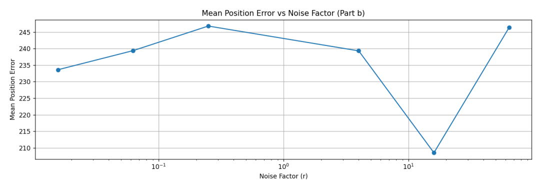

Figure 2: Mean position error vs Noise factor

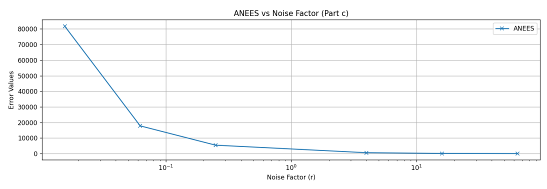

Figure 3: Anees vs Noise factor

4.2 PF Analysis (pf_analysis.py)

4.2.1 Mean Position Error vs Noise Factor (Parts b and d)

- Part b: Mean position error decreases as the noise factor increases, similar to EKF behavior.
- Very high noise factors may degrade accuracy and increase error.
- Part d: Increasing the number of particles reduces the mean position error, especially at higher noise levels.

4.2.2 ANEES vs Noise Factor (Parts c and d)

- Part c: ANEES decreases with increasing noise factor and stabilizes around 0.
- Part d: Increasing particle count reduces ANEES, particularly at lower noise levels.
- This suggests that more particles improve the filter's consistency.

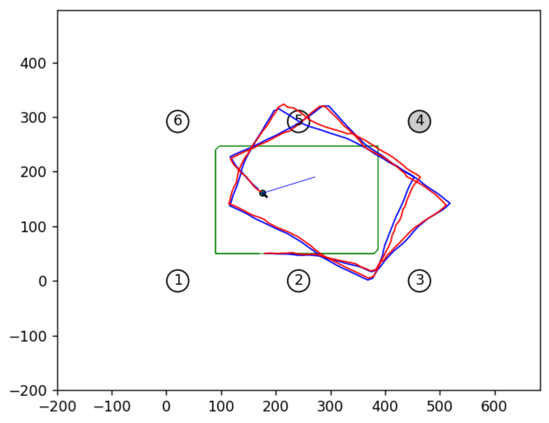

Figure 4: Path

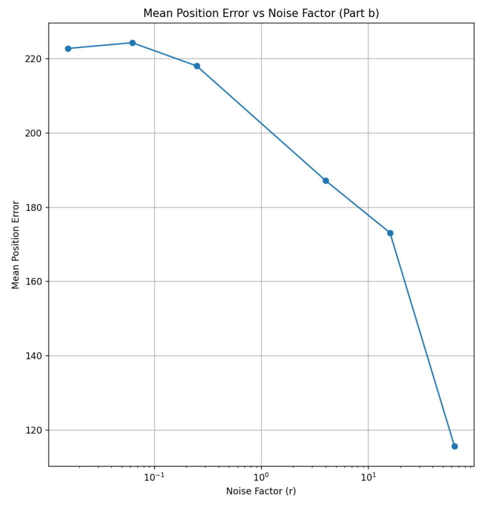

Figure 5: Mean position error vs Noise factor

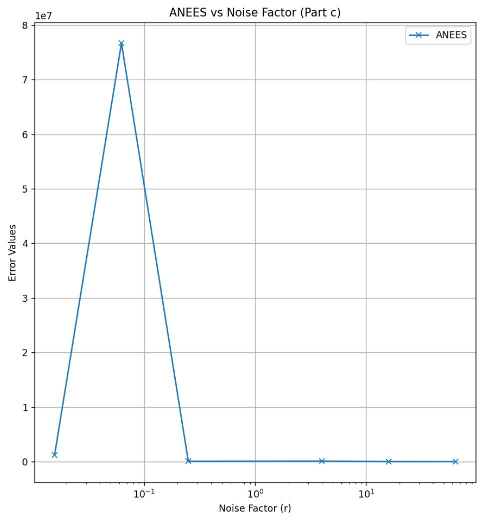

Figure 6: Anees vs Noise factor

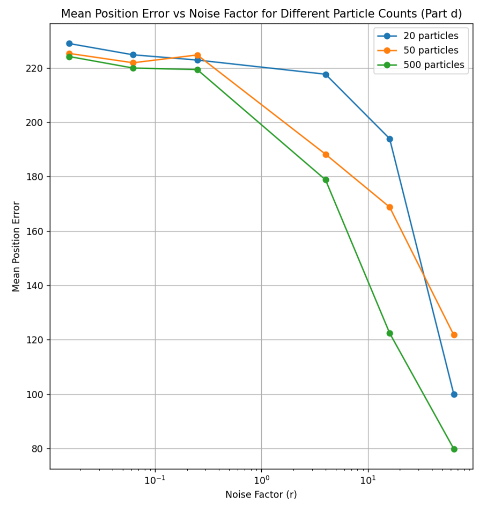

Figure 7: Mean position error vs Noise factor

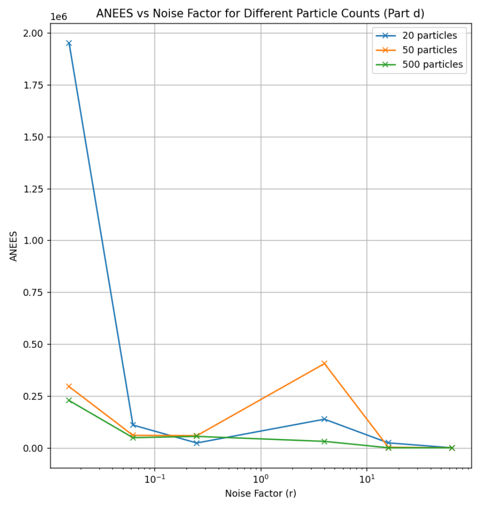

Figure 8: Anees vs Noise factor

4.3 Comparison of EKF and PF

- Both EKF and PF show similar trends in mean position error and ANEES as noise factor varies.
- PF generally performs better at higher noise levels due to its ability to handle non-linear and non-Gaussian noise.
- PF performance improves with higher particle count.
- EKF performance is primarily influenced by the accuracy of the noise model.