# Homework 3

## ASEN 6519 Cooperative Information Gathering

## Assigned: Wednesday 2/20/25

## Due: Wednesday 2/27/25, 1 PM

This assignment considers coordinated tracking of a single target by one or more unmanned aircraft. The unmanned aircraft have range and bearing sensors.

- Let $\mathbf{x}_k$ be the target state consisting of the *two dimensional* position of the target in the inertial coordinate system at discrete time $k$ and the two dimensional velocity of the target.
- Let the target move with motion model $\mathbf{x}_{k+1} = \mathbf{f}_t(\mathbf{x}_k, \mathbf{w}_k)$ where $\mathbf{w}_k$ is process noise.
- Let $\mathbf{x}_k^i = [x_k^i \quad y_k^i \quad \psi^i]^T$ be the pose of the $i$th sensor where $\psi_k^i$ is the heading of the sensor platform. Let there by $N$ total sensors, where the number of sensors will vary between problems.
- Let $\mathbf{u}_k^i$ be the control input of the $i$th sensor that drives the sensor motion model $\mathbf{x}_{k+1}^i = \mathbf{f}_s(\mathbf{x}_k^i, \mathbf{u}_k^i)$, where each sensor has the same model.
- Let $\mathbf{z}_k^i = \mathbf{h}(\mathbf{x}_k, \mathbf{x}_k^i, \mathbf{v}_k^i)$ be the measurement taken by the $i$th sensor at time k, with measurement model $\mathbf{h}(\mathbf{x}_k, \mathbf{x}_k^i, \mathbf{v}_k^i)$ where $\mathbf{v}_k^i$ is the measurement noise. Every sensor has the same measurement model. Note, the measurement model is a function of the target state and the sensor state.

### Sensor Motion Model

The sensor moves according to a "kinematic unicycle model" such that

$$\begin{aligned} x_{k+1}^i &= x_k^i + u_k^{i,1} \cos(\psi_{k+1}^i)\Delta T \\ y_{k+1}^i &= y_k^i + u_k^{i,1} \sin(\psi_{k+1}^i)\Delta T \\ \psi_{k+1}^i &= \psi_k^i + u_k^{i,2}\Delta T \end{aligned} \tag{1}$$

where $\Delta T$ is the sample time, $V_{min} \leq u_k^{i,1} \leq V_{max}$ is the sensor speed constrained by minimum and maximum values, and $|u_k^{i,2}| \leq \omega_{max}$ is the turn rate constrained by a maximum value.

### Target Model

The target moves according to a "constant velocity model" where the process noise is the uncertain acceleration. The target model is therefore

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} + \begin{bmatrix} \Delta T^2/2 & 0 \\ 0 & \Delta T^2/2 \\ \Delta T & 0 \\ 0 & \Delta T \end{bmatrix} \begin{bmatrix} w_{x,k} \\ w_{y,k} \end{bmatrix} \quad (2)$$

where $\mathbf{w}_k = [w_{x,k} \quad w_{y,k}]^T$ is the zero-mean Gaussian process noise with covariance matrix $\mathbf{Q}_k$.

**Measurement Equation**

The range-bearing sensor has measurement model

$$\mathbf{z}_k^i = \begin{bmatrix} r_k^i \\ \beta_k^i \end{bmatrix} = \begin{bmatrix} \sqrt{(x_k - x_k^i)^2 + (y_k - y_k^i)^2} \\ \tan^{-1} \dfrac{y_k - y_k^i}{x_k - x_k^i} \end{bmatrix} + \begin{bmatrix} v_{r,k}^i \\ v_{\beta,k}^i \end{bmatrix} \quad (3)$$

where $\mathbf{v}_k = [v_{r,k} \quad v_{\beta,k}]^T$ is the zero-mean Gaussian measurement noise with covariance matrix $\mathbf{R}_k$.

## Problem 1

Write the following Matlab functions:

`sensor_state_next = SensorMotionModel(sensor_state_curr, sensor_control, delta_t)`
This function implements the sensor motion model. The inputs to the file are the current state (3 by 1), control input (2 by 1) and scalar sample time. It returns the 3 by 1 sensor state at the next time step.

`target_state_next = ConstantVelocityTargetModel(target_state_curr, process_noise, delta_t)`
This function implements the constant-velocity target motion model. The inputs to the file are the current target state (4 by 1), process noise (2 by 1) and scalar sample time. It returns the 4 by 1 target state at the next time step.

`sensor_meas = RangeBearingSensor(target_state, sensor_state, sensor_noise)`
This function implements the range-bearing sensor model. The inputs to the file are the current target state (4 by 1), sensor state (3 by 1) and sensor noise (2 by 1). It returns the 2 by 1 sensor measurement.

`H = RangeBearingSensorJacobian(target_state, sensor_state)`
This function returns the 2 by 4 Jacobian matrix (partial derivatives) of the sensor model. The inputs to the file are the current target state (4 by 1) and sensor state (3 by 1).

## Problem 2

Simulate the following scenario (all units SI: m, m/s, rad, etc). Generate the sensor path, target path, and sensor measurements. Save all data for processing in Problem 3.

Note: The simulation includes random process noise and sensor noise. You are precomputing the target path and sensor measurements so Problem 3 and Problem 4 use the exact same data so you can compare the filters.

**Table 1:** Simulation parameters

| | |
|---|---|
| Sample time $\Delta T$ | 1 s |
| Final Time | 200 s |
| Initial target state $\mathbf{x}_0$ | $[2000,\ 0,\ 12,\ 5.5]^T$ |
| Process noise covariance $\mathbf{Q}$ | $0.05 \cdot \mathbf{I}$ |
| Initial sensor state $\mathbf{x}_0^1$ | $[0,\ 0,\ 60\pi/180]^T$ |
| Sensor control trajectory | $u_k^{1,1} = 20, \forall k$; $u_k^{1,2} = -0.05, 89 \leq k \leq 110$ otherwise $u_k^{2,1} = 0$; |
| Sensor noise covariance $\mathbf{R}$ | $\begin{bmatrix} 5^2 & 0 \\ 0 & (10 * \pi/180)^2 \end{bmatrix}$ |

Create Figure 1 and plot the sensor path and the target path on it, using different colors for each path.

## Problem 3

Implement an Extended Kalman Filter to estimate the target motion. You may implement the filter however you like (hard coded into an iterative loop or as a function like I showed in class). Assume you have prior information that tells you the target state has a normal distribution with mean $\mathbf{x}_0 = [2400; -200; 8; 8.5]^T$ and covariance $\mathbf{P}_0 = [100^2\mathbf{I} \quad \mathbf{0}; \mathbf{0} \quad 10^2\mathbf{I}]$.

- Add the estimated target path (in a new color) to Figure 1.
- Create Figure 2 that includes 4 subplots, one for each target state, of the estimate error versus time and the $3-\sigma$ bounds from the covariance matrix.

## Problem 4

Implement a Sigma Point Filter to estimate the target motion. You may implement the filter however you like (hard coded into an iterative loop or as a function like I showed in class). Assume you have prior information that tells you the target state has a normal distribution with mean $\mathbf{x}_0 = [2400; -200; 8; 8.5]^T$ and covariance $\mathbf{P}_0 = [100^2\mathbf{I} \quad \mathbf{0}; \mathbf{0} \quad 10^2\mathbf{I}]$.

- Add the estimated target path in a different color to Figure 1.
- Add plots for the estimate error to Figure 2 usung the same color as the estimated target path from the previous part.

## Problem 5

Write the following Matlab function:

```
info_cost = InfoCost(control_seq, sensor_state_curr, target_state_est,
target_state_cov, delta_t)
```

This function uses the Extended Kalman Filter equations to predict the covariance matrix of the target. The input is a control sequence of the sensor turn rate. Based on the length of this input, iteratively compute the covariance matrix. In order to linearize the measurement equations, predict the target motion from the estimate assuming no additional process noise (assume $\mathbf{w}_{k_i} = \mathbf{0}$).

Let $N_p =$ `length(control_seq)` and $\mathbf{P}_{k|k}$ be the predicted covariance matrix at time $k$, then the information cost $J_{info}$ returned by the function should be the sum of the log of the determinant of the covariance matrix at each time step:

$$J_{info} = \sum_{i=k+1}^{k+N_p} \log |\mathbf{P}_{i|i}|.$$

This function does not need to be generalized like the example code I showed in class. You may hard code in the use of the sensor and target models.

## Problem 6

Use the Matlab function `fmincon` to perform the following optimizations to minimize the information cost. The initial conditions for all scenarios are the sensor and target states, and state estimates, given in Problem 2 and Problem 3. For the purposes of planning use a fixed speed $u_k^{1,1} = v_0 = 20$ m/s and a maximum turn rate of $|u_k^{2,1}| \leq \omega_{max} = 0.2$ rad/sec. Each simulation should end with a trajectory of length $t_{final} = 100$ s. (If the simulation starts at time $t = 0$ and has $\Delta T = 1$ s, then the final solution will have 101 elements.)

- Greedy offline: Use $\Delta T = 1$s and $N_p = 1$. Use the result of this simulation to determine the next position of the sensor and repeat. Do not update the actual target estimate between runs, use the predictions generated by assuming no process noise on the target and the predicted information matrix. Based on the resulting final trajectory, simulate the sensor measurements and use the Sigma Point filter to generate an estimate of the target state.

- Greedy online: Use $\Delta T = 1$s and $N_p = 1$. Use the result of this simulation to determine the next position of the sensor and repeat. Run the Sigma Point filter after each step. Use the resulting output of the filter to initialize the optimization at the next step.

- Receding horizon offline: Use $\Delta T = 1$s, $N_p = 10$, and $N_c = 1$. Use the result of this simulation to determine the next position of the sensor and repeat. Do not update

the actual target estimate between runs, use the predictions generated by assuming no process noise on the target and the predicted information matrix. Based on the resulting final trajectory, simulate the sensor measurements and use the Sigma Point filter to generate an estimate of the target state.

- Receding horizon offline: Use $\Delta T = 1$s, $N_p = 10$, and $N_c = 1$. Use the result of this simulation to determine the next position of the sensor and repeat. Run the Sigma Point filter after each step. Use the resulting output of the filter to initialize the optimization at the next step.

- Full optimal offline: Use $\Delta T = 1$s and $N_p = 100$. Run the simulation with this path and use the Sigma Point filter to estimate performance.

Generate plots that compare the five different paths. Show the different paths themselves as well as the performance of the Sigma Point filter using each path. Comment on the value of the different control appoaches.

## Hints

- Here is the syntax for using fmincon:

  ```
  [opt_seq, fval] = fmincon(@(x)InfoCost(x, sensor_state_curr, target_state_est,
  target_state_cov, delta_t)),seq0,[],[],[],[],lb,ub,[],options)
  ```

- In the example above the vectors `ub = omega_max*ones(n_p,1)` and `lb = - ub`.

- It might be helpful to write a function that takes in the target path and sensor path, and runs the full estimation problem. You can call this after each plan is done.

- It might also be helpful to write separate functions that perform the open loop and closed loop plans. If written well, you can have the planning horizon be one of the function inputs and then use the same function for Problem 6a and Problem 6c (and the other function for Problem 6b and Problem 6d).

- My results for Problem 3 and Problem 4 are presented below. Yours should look close. (Your measurements will be different so the results will not be identical).

## SUBMISSION

For this assignment only submit the requested figures. Like previous assignments, grading will be minimal based on if you complete the problems and if they are correct.

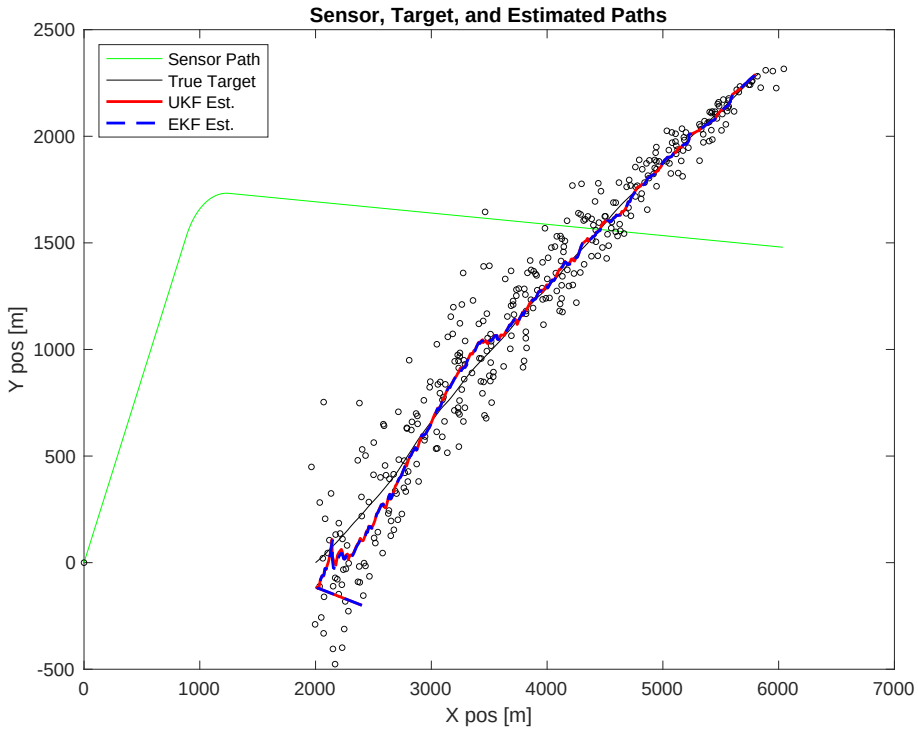


**Figure 1:** Path Results for Problem 1 - 4.

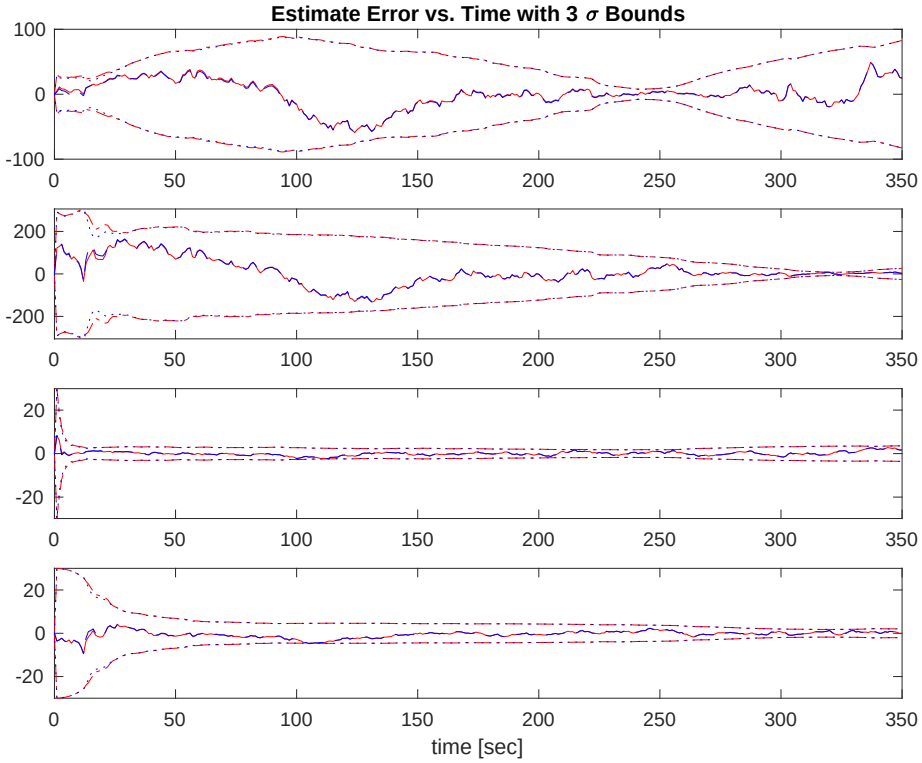


**Figure 2:** Estimator Results for Problem 3 - 4.