ECE4250-5250: DSP and Statistical Inference

Target Pursue

Fall, 2022

This project simulates a target pursue scenario where a pursuer aims to capture possibly multiple targets in a two-dimensional plane. The project includes two parts:

P1 Target state estimation via Kalman filtering.

P2 Multi-target pursue.

In the lectures, we have covered the Kalman filtering technique to estimate the system state—the positions and velocities of targets. Capturing targets (P2) involves control decisions of the pursuer, which we do not cover in this course. Here, you are free to try different strategies.

The project includes theoretical analysis, numerical calculations, and scenario simulations. The simulation software assumed in this project is MATLAB. If you insist on using a different simulation Language, you will need to provide a validation program. Contact the instructor for details.

Strict adherence to academic integrity is expected. Discussions with fellow students in this class are allowed and encouraged. You must, however, write your own solution, produce your own code, and acknowledge your discussions with other students.

The project is due on **December 16, 2022 (11:59 PM) on canvas.** You need to submit one zipped folder that includes the following:

- **Project report:** There is no need to write an elaborate report. Start each problem on a new page, and present your derivation and reasoning, tables, and figures. Discussions about your findings and comments on worthy features of your solution are highly valued.
- **Code:** Submit the MATLAB script for each subproblem. Name your file according to the section and problem numbers: `P1.1.x` means that it is a file related to Section 1 and problem 1.

1 Target Mobility and Observation Models

Figure 1 below illustrates a target pursue problem within a rectangular box $[0, B_x] \times [0, B_y]$ for the multi-target scenario. All targets move in approximately constant speeds with small random deviations. When a target reaches the boundary, it bounces (reflected perfectly) off the boundary.

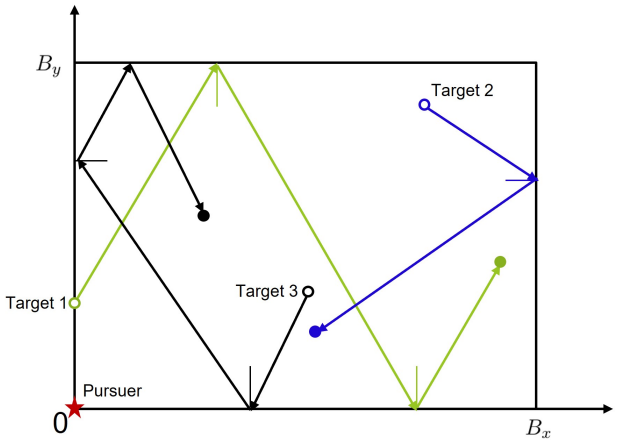

Figure 1: Multitarget pursue. The initial positions of the targets are labeled by (hollow) circles.

We first derive a discrete-time state-space model involving N targets in the form of

$$\begin{aligned} \boldsymbol{s}[n+1] &= \boldsymbol{A}\boldsymbol{s}[n] + \boldsymbol{x}[n] + \boldsymbol{u}[n] \\ \boldsymbol{y}[n] &= \boldsymbol{C}\boldsymbol{s}[n] + \boldsymbol{w}[n], \end{aligned} \tag{1}$$

for $n = 0, 1, \cdots$, where $\boldsymbol{s}[n]$ is the state vector involving N targets, $\boldsymbol{u}[n]$ and $\boldsymbol{w}[n]$ the additive process and observation noise, and $\boldsymbol{x}[n]$ the target inputs that reflect a target back inside the rectangular box whenever it reaches the boundary. The pursuer starts at the origin.

Target mobility model. The state vector $\boldsymbol{s}_i[n]$ of target i at time n includes the horizontal/vertical position vector $\boldsymbol{z}_i[n] = \left[z_i^{\mathsf{H}}[n], z_i^{\mathsf{V}}[n]\right]^\top$ and the horizontal/ vertical velocity vector $\boldsymbol{v}_i[n] = \left[v_i^{\mathsf{H}}[n], v_2^{\mathsf{V}}[n]\right]^\top$. Specifically,

$$\boldsymbol{s}_i[n] = \begin{bmatrix} \boldsymbol{z}_i[n] \\ \boldsymbol{v}_i[n] \end{bmatrix} = \begin{bmatrix} z_i^{\mathsf{H}}[n] \\ z_i^{\mathsf{V}}[n] \\ v_i^{\mathsf{H}}[n] \\ v_i^{\mathsf{V}}[n] \end{bmatrix} \quad \begin{aligned} z_i^{\mathsf{H}}[n]: &\ \text{horizontal position of target } i, \\ z_i^{\mathsf{V}}[n]: &\ \text{vertical position of target } i, \\ v_i^{\mathsf{H}}[n]: &\ \text{horizontal velocity of target } i, \\ v_i^{\mathsf{V}}[n]: &\ \text{vertical velocity of target } i. \end{aligned}$$

We assume that targets are all within a rectangular box: $x_i^{\mathsf{H}}[n] \in [0, B_x]$ and $x_i^{\mathsf{V}}[n] \in [0, B_y]$. The velocities $(v_i^{\mathsf{H}}[n], v_i^{\mathsf{V}}[n])$ are measured in meters per unit step.

The discrete-time state-space equation for target i is given by

$$\boldsymbol{s}_i[n] = \boldsymbol{A}_i\boldsymbol{s}_i[n] + \boldsymbol{x}_i[n] + \boldsymbol{u}_i[n], \tag{2}$$

where $\boldsymbol{u}_i[n]$ is the Gaussian *process noise* and $\boldsymbol{x}_i[n]$ the target input that reflects the state back inside the rectangular box whenever the target reaches the boundary.

At time n, suppose that the target i is at position $\boldsymbol{z}_i[n]$ moving at the speed of $\boldsymbol{v}_i[n]$. The next state $\boldsymbol{s}_i[n+1]$ is given by

$$\begin{aligned}\boldsymbol{s}_i[n+1] &= \begin{bmatrix} \boldsymbol{z}_i[n+1] \\ \boldsymbol{v}_i[n+1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_i[n] \\ \boldsymbol{v}_i[n] \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_i[n] \\ \boldsymbol{0} \end{bmatrix} + \boldsymbol{x}_i[n] + \boldsymbol{u}_i[n] \\ &= \underbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{A}_i} \underbrace{\begin{bmatrix} \boldsymbol{z}_i[n] \\ \boldsymbol{v}_i[n] \end{bmatrix}}_{\boldsymbol{s}_i[n]} + \boldsymbol{x}_i[n] + \boldsymbol{u}_i[n] \\ &= \boldsymbol{A}_i \boldsymbol{s}_i[n] + \boldsymbol{x}_i[n] + \boldsymbol{u}_i[n]\end{aligned}$$

where $\boldsymbol{I}$ is the 2×2 identity matrix, $\boldsymbol{u}_i[n] \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_i[n])$ the additive Gaussian noise with possibly varying covariance $\boldsymbol{R}_i[n]$, and $\boldsymbol{x}_i[n]$ is the corrective input of the target when it runs into the boundary.

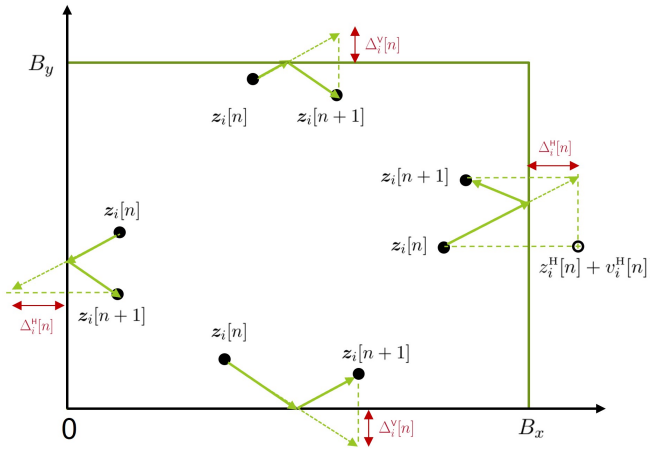

Figure 2: Target state near the boundary.

Figure 2 shows the effects of corrective action of $\boldsymbol{x}_i[n]$. When $z_i^{\mathrm{H}}[n] + v_i^{\mathrm{H}}[n] > B_x$ beyond the upper boundary at B_x (See the case near the right boundary of Figure 2), the target input reduces the the horizontal position by $2\Delta_i^{\mathrm{H}}[n]$. The same applies also to the lower horizon boundary at 0 and to the upper and lower vertical boundaries. The corrective action input $\boldsymbol{u}_i[n]$ also reverse the direction of the speed when the state update exceeds the boundary.

Taking into account all possible situations at the boundary, let

$$\begin{aligned}\Delta_i^{\mathrm{H}}[n] &= \min\{0, z_i^{\mathrm{H}}[n] + v_i^{\mathrm{H}}[n]\} + \max\{0, z_i^{\mathrm{H}}[n] + v_i^{\mathrm{H}}[n] - B_x\} \\ \Delta_i^{\mathrm{V}}[n] &= \min\{0, z_i^{\mathrm{V}}[n] + v_i^{\mathrm{V}}[n]\} + \max\{0, z_i^{\mathrm{V}}[n] + v_i^{\mathrm{V}}[n] - B_y\}\end{aligned}$$

be amount the updated state exceeding the horizontal/vertical boundaries, respectively. Note that when the state update is within the boundary, $\Delta_i^{\mathrm{H}}[n]$ and $\Delta_i^{\mathrm{V}}[n]$ are both zero. And

when the state update is beyond boundary, only one of the two terms in $\Delta_i^{\mathrm{H}}[n]$ (and $\Delta_i^{\mathrm{V}}[n]$) is non-zero. The the corrective control input $\boldsymbol{x}_i[n]$ is given by

$$\boldsymbol{x}_i[n] = -2 \begin{bmatrix} \Delta_i^{\mathrm{H}}[n] \\ \Delta_i^{\mathrm{V}}[n] \\ \mathsf{sign}(\Delta_i^{\mathrm{H}}[n]) v_i^{\mathrm{H}}[n] \\ \mathsf{sign}(\Delta_i^{\mathrm{V}}[n]) v_i^{\mathrm{V}}[n] \end{bmatrix} \quad \text{where} \quad \mathsf{sign}(a) := \begin{cases} -1 & a < 0 \\ 0 & a = 0 \\ 1 & a > 0. \end{cases}$$

Note that $\boldsymbol{x}_i[n] = 0$ whenever the state update is within the boundary.

The state involving N targets is the (column) vector is given by $\boldsymbol{s}[n]^\top = [\boldsymbol{s}_1^\top, \cdots, \boldsymbol{s}_N^\top]$ with state transition $\boldsymbol{s}[n+1] = \boldsymbol{A}\boldsymbol{s}[n] + \boldsymbol{x}[n] + \boldsymbol{u}[n]$ given by

$$\underbrace{\begin{bmatrix} \boldsymbol{s}_1[n+1] \\ \vdots \\ \boldsymbol{s}_N[n+1] \end{bmatrix}}_{\boldsymbol{s}[n+1]} = \underbrace{\begin{bmatrix} \boldsymbol{A}_1 & & \\ & \ddots & \\ & & \boldsymbol{A}_N \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{s}_1[n] \\ \vdots \\ \boldsymbol{s}_N[n] \end{bmatrix}}_{\boldsymbol{s}[n]} + \underbrace{\begin{bmatrix} \boldsymbol{x}_1[n+1] \\ \vdots \\ \boldsymbol{x}_N[n+1] \end{bmatrix}}_{\boldsymbol{x}[n]} + \underbrace{\begin{bmatrix} \boldsymbol{u}_1[n+1] \\ \vdots \\ \boldsymbol{u}_N[n+1] \end{bmatrix}}_{\boldsymbol{u}[n]} \tag{3}$$

where $\boldsymbol{u}[n] \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}[n])$, and $\boldsymbol{R}[n] = \mathsf{diag}(\boldsymbol{R}_1[n], \cdots, \boldsymbol{R}_N[n])$.

Target observation model. We assume that the pursuer have noisy measurements $\boldsymbol{y}_i[n]$ of the position $\boldsymbol{x}_i[n]$ of target i in the form

$$\begin{aligned} \boldsymbol{y}_i[n] = \boldsymbol{z}_i[n] + \boldsymbol{v}_i[n] &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{\boldsymbol{C}_i} \boldsymbol{s}_i[n] + \boldsymbol{w}_i[n] \\ &= \boldsymbol{C}_i \boldsymbol{s}_i[n] + \boldsymbol{w}_i[n] \end{aligned}$$

where $\boldsymbol{w}_i[n] \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_i[n])$ is an additive Gaussian noise with possibly varying covariance $\boldsymbol{Q}_i[n]$. The observation model involving all targets is given by

$$\boldsymbol{y}[n] = \begin{bmatrix} \boldsymbol{y}_1[n] \\ \vdots \\ \boldsymbol{y}_N[n] \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{C}_1 & & \\ & \ddots & \\ & & \boldsymbol{C}_N \end{bmatrix}}_{\boldsymbol{C}} \underbrace{\begin{bmatrix} \boldsymbol{s}_1[n] \\ \vdots \\ \boldsymbol{s}_N[n] \end{bmatrix}}_{\boldsymbol{s}[n]} + \underbrace{\begin{bmatrix} \boldsymbol{w}_1[n] \\ \vdots \\ \boldsymbol{w}_N[n] \end{bmatrix}}_{\boldsymbol{w}[n]}. \tag{4}$$

Assuming independent observation noise, we have $\boldsymbol{w}[n] \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}[n])$ with block diagonal covariance matrix $\boldsymbol{Q}[n] = \mathsf{diag}(\boldsymbol{Q}_1[n], \cdots, \boldsymbol{Q}_N[n])$.

With (3)-(4), we have derived the compete state-space model (1). A MATLAB script that generates actual trajectories and measurements will be provided on Canvas.

Note that all targets operate independent of each other, and there is no coupling among target observations. Even though we have the overall system state equation (1), we can perform state estimation of each target independently.

2 Part I: Target State Estimation

Part I considers target state estimation based on measurements $\boldsymbol{y}_{[0:n]} := (\boldsymbol{y}[n], \boldsymbol{y}[n-1], \cdots, \boldsymbol{y}[0])$. Recall the notations and equations involved in Kalman filtering that produces the linear minimum mean squared error (LMMSE) state estimates. (Under the Gaussian model, Kalman filter gives the MMSE state estimates.)

Observation data:

- $\boldsymbol{y}_{[0:n]}$: Observation up to time n. $\boldsymbol{y}_{[0:n]} := (\boldsymbol{y}[n], \boldsymbol{y}[n-1], \cdots, \boldsymbol{y}[0])$

Kalman state and output predictions:

- $\hat{\boldsymbol{s}}_{n|n-1}$: the LMMSE estimate (prediction) of $\boldsymbol{s}[n]$ from $\boldsymbol{y}_{[0,n-1]}$.
- $\boldsymbol{\Sigma}_{n|n-1}$:the mean squared error (MSE) of the state prediction.

$$\boldsymbol{\Sigma}_{n|n-1} := \mathbb{E}\Big[\Big(\boldsymbol{s}[n] - \hat{\boldsymbol{s}}_{n|n-1}\Big)\Big(\boldsymbol{s}[n] - \hat{\boldsymbol{s}}_{n|n-1}\Big)^\top \Big| \boldsymbol{y}_{[0:n-1]}\Big].$$

- $\hat{\boldsymbol{y}}_{n|n-1}$: the LMMSE estimate of Y_n from $\boldsymbol{y}_{[0,n-1]}$

$$\hat{\boldsymbol{y}}_{n|n-1} = \boldsymbol{C}\hat{\boldsymbol{s}}_{n|n-1}.$$

Kalman state estimation:

- $\hat{\boldsymbol{s}}_{n|n}$: the LMMSE estimate of $\boldsymbol{s}[n]$ from $\boldsymbol{y}_{[0,n]}$.
- $\boldsymbol{\Sigma}_{n|n}$:the mean squared error (MSE) of the state estimation.

$$\boldsymbol{\Sigma}_{n|n} := \mathbb{E}\Big[\Big(\boldsymbol{s}[n] - \hat{\boldsymbol{s}}_{n|n}\Big)\Big(\boldsymbol{s}[n] - \hat{\boldsymbol{s}}_{n|n}\Big)^\top | \boldsymbol{y}_{[0:n]}\Big]$$

- $\boldsymbol{K}_n$: The Kalman gain matrix at time n.

Kalman updates

- Initialization: Assume $\boldsymbol{s}_0 \sim \mathcal{N}(\mu_0, \boldsymbol{P}_0)$. We have

$$\begin{aligned}\hat{\boldsymbol{s}}_{0|-1} &= \mu_0, \quad \boldsymbol{\Sigma}_{0|-1} = \boldsymbol{P}_0\\ \hat{\boldsymbol{y}}_{0|-1} &= \boldsymbol{C}\hat{\boldsymbol{s}}_{0|-1}.\end{aligned}$$

- Measurement updates: For $n = 0, 1, \cdots,$

$$\begin{aligned}\boldsymbol{K}_n &:= \boldsymbol{\Sigma}_{n|n-1}\boldsymbol{C}^\top\big(\boldsymbol{C}\boldsymbol{\Sigma}_{n|n-1}\boldsymbol{C}^\top + \boldsymbol{Q}[n]\big)^{-1}\\ \hat{\boldsymbol{s}}_{n|n} &= \hat{\boldsymbol{s}}_{n|n-1} + \boldsymbol{K}_n(\boldsymbol{y}[n] - \hat{\boldsymbol{y}}_{n|n-1})\\ \boldsymbol{\Sigma}_{n|n} &= \boldsymbol{\Sigma}_{n|n-1} - \boldsymbol{K}_n\boldsymbol{C}\boldsymbol{\Sigma}_{n|n-1}\end{aligned}$$

- Time updates: For $n = 1, \cdots,$

$$\begin{aligned}\hat{\boldsymbol{s}}_{n+1|n} &= \boldsymbol{A}\hat{\boldsymbol{s}}_{n|n}, \quad \hat{\boldsymbol{y}}_{n+1|n} = \boldsymbol{C}\hat{\boldsymbol{s}}_{n+1|n}\\ \boldsymbol{\Sigma}_{n+1|n} &= \boldsymbol{A}\boldsymbol{\Sigma}_{n|n}\boldsymbol{A}^\top + \boldsymbol{R}[n] = \boldsymbol{A}\big(\boldsymbol{\Sigma}_{n|n-1} - \boldsymbol{K}_n\boldsymbol{C}\boldsymbol{\Sigma}_{n|n-1}\big)\boldsymbol{A}^\top + \boldsymbol{R}[n]\end{aligned}$$

Problem 1: Single target state estimation without boudaries (50 points) We first consider the simple case when there is a single target and there is no boundary, *i.e.,* $B_x = B_y = \infty$. In this case, the target control $\boldsymbol{x}[n] = \mathbf{0}$. The parameters of the state-space model are given by

$$\boldsymbol{z}[0] = \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \quad \boldsymbol{v}[0] = \begin{bmatrix} 0.1 \\ 0.05 \end{bmatrix}, \quad \boldsymbol{R}[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}[n] = \sigma_W^2 \boldsymbol{I}.$$

P1.1 For $n = 0, 1, \cdots, 250$, compute $\boldsymbol{K}_n$ for $\sigma_W^2 = 0.005, 0.05, 0.5$ and generate four separate figures:

(a) Fig. 1.1a: $K_n(1,1)$ for $\sigma_W^2 = 0.005, 0.05, 0.5$

(b) Fig. 1.1b: $K_n(1,2)$ for $\sigma_W^2 = 0.005, 0.05, 0.5$

(c) Fig. 1.1c: $K_n(2,1)$ for $\sigma_W^2 = 0.005, 0.05, 0.5$

(d) Fig. 1.1d: $K_n(2,2)$ for $\sigma_W^2 = 0.005, 0.05, 0.5$

Comment on the trend of these figures.

P1.2 For $n = 0, 1, \cdots, 250$ and $\sigma_W = 0.05$, compute the MSE of the Kalman estimate $\boldsymbol{\Sigma}_{n|n}$ and plot in the same figure of the diagonals of the error covariance matrix

$$\boldsymbol{\Sigma}_n(1,1), \boldsymbol{\Sigma}_n(2,2), \boldsymbol{\Sigma}_n(3,3), \boldsymbol{\Sigma}_n(4,4) \text{ vs. } n$$

P1.3 Assume $\boldsymbol{\Sigma}_{n|n}$ converges and $\sigma_W = 0.05$. Solve the asymptotic MSE $\boldsymbol{\Sigma}_{\infty|\infty}(1,1), \boldsymbol{\Sigma}_{\infty|\infty}(2,2)$ from the covariance update equation and compare them with $\boldsymbol{\Sigma}_{250}(1,1)$ and $\boldsymbol{\Sigma}_{250}(2,2)$. Do you observe convergence?

P1.4 For $n = 0, 1, \cdots, 250$ and $\sigma_W = 0.05$, generate one realization of $(\boldsymbol{s}[n], \boldsymbol{y}[n])$ and compute the position estimate $\hat{\boldsymbol{x}}_{n|n}$. Plot in the same figure the actual state trajectory and estimated state trajectory.

P1.5 Assume $\sigma_W = 0.05$. Let the total MSE of the Kalman state estimate be

$$\mathcal{E}_{n|n} := \sum_{i=1}^{4} \boldsymbol{\Sigma}_{n|n}(i,i).$$

Generate 500 Monte Carlo realizations of the state and observation trajectories. Compute the empirical total MSE

$$\hat{\mathcal{E}}_{n|n} = \frac{1}{500} \sum_{k=1}^{500} (||\hat{\boldsymbol{s}}_{n|n}^{(k)} - \boldsymbol{s}^{(k)}[n]||^2)$$

where k is the Monte Carlo simulation index, $(\boldsymbol{s}^{(k)}[n])$ the state sequence generated in the kth Monte Carlo run, and $\hat{\boldsymbol{s}}_{n|n}^{(k)}$ the state estimate in the kth Monte Carlo run. Plot in the same figure $\mathcal{E}_{n|n}$ and $\hat{\mathcal{E}}_{n|n}$ vs. n.

Problem 2: Single target state estimation with boudaries (20 points) Now we assume that the target have to stay with the rectangular box $[0, 20] \times [0, 20]$. In this case, the target control $\boldsymbol{x}[n]$ may not be zero.

Strictly speaking, the model deviates from the linear state space model assumed in the Kalman filter derivation where the external input $\boldsymbol{x}[n]$ does not depend on the state as our model assumes. Nonetheless, since $\boldsymbol{x}[n]$ is only nonzero when the target runs into the boundary, we ignore its effect in the Kalman state estimation.

The parameters of the state-space model are the same as in Problem 1.

P2.1 Repeat P1.4.

P2.2 Repeat P1.5.

Problem 3: Multi-target state estimation with boudaries (10 points) Now we assume that the target have to stay with the rectangular box $[0, 20] \times [0, 20]$. The parameters of the state-space model are given by

$$\boldsymbol{z}_1[0] = \begin{bmatrix} 0 \\ 5 \end{bmatrix}, \quad \boldsymbol{v}_1[0] = \begin{bmatrix} 0.1 \\ 0.05 \end{bmatrix}, \quad \boldsymbol{R}_1[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}_1[n] = 0.05\boldsymbol{I}.$$

$$\boldsymbol{z}_2[0] = \begin{bmatrix} 10 \\ 0 \end{bmatrix}, \quad \boldsymbol{v}_2[0] = \begin{bmatrix} -0.1 \\ 0.05 \end{bmatrix}, \quad \boldsymbol{R}_2[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}_2[n] = 0.05\boldsymbol{I}.$$

$$\boldsymbol{z}_3[0] = \begin{bmatrix} 15 \\ 15 \end{bmatrix}, \quad \boldsymbol{v}_3[0] = \begin{bmatrix} -0.1 \\ -0.05 \end{bmatrix}, \quad \boldsymbol{R}_3[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}_3[n] = 0.05\boldsymbol{I}.$$

P3.1 Repeat P1.5.

3 Part II: Target Pursue

We now consider the problem of target pursue with a single pursuer whose state is defined by its position $\boldsymbol{\alpha}[n] = [\alpha^{\mathsf{H}}[n], \alpha^{\mathsf{V}}[n]]^\top$. The state evolution equation is given by

$$\boldsymbol{\alpha}[n+1] = \boldsymbol{\alpha}[n] + \boldsymbol{\nu}[n], \quad \boldsymbol{\alpha}[0] = [0,0]^\top.$$

where $\boldsymbol{\nu}[n] = [\nu^{\mathsf{H}}[n], \nu^{\mathsf{V}}[n]]$ is the speed vector of the pursuer to be determined at each time step. We assume that the *maximum speed* of the pursuer is 0.2 in both horizontal and vertical directions, *i.e.,* at each step, the pursuer, can choose to move at the speed $\boldsymbol{\nu}[n]$ satisfying $|\nu^{\mathsf{H}}[n]|, |\nu^{\mathsf{V}}[n]| \leq 0.2$ for all n.

We assume that the target i is captured at time n if the target is 0.2 (Euclidean) distance away from the pursuer, *i.e.,* $||\boldsymbol{z}_i[n] - \boldsymbol{\alpha}[n]|| \leq 0.2$, as shown in Figure 3.

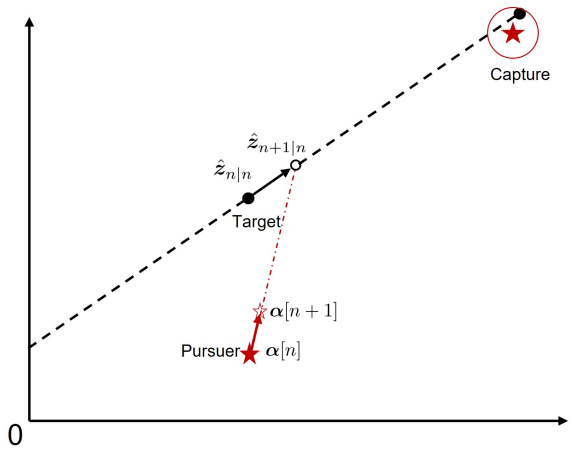

Figure 3: Target pursue based on one-step prediction.

Problem 4: Single target pursue without boundary via one-step predictive pursue (20 points) Consider the setting of Problem 1 and a *one-step predictive pursue* strategy that, at time n, the target moves at the maximum speed in the direction of the predicted target state $\hat{\boldsymbol{z}}_{n+1|n}$ at $n+1$, as shown in Figure 3.

Assume the following parameters:

$$\boldsymbol{z}[0] = \begin{bmatrix} R\cos(\Theta) \\ R\sin(\Theta) \end{bmatrix}, \quad \boldsymbol{v}[0] = \begin{bmatrix} 0.1 \\ 0.05 \end{bmatrix}, \quad \boldsymbol{R}[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}[n] = 0.05\boldsymbol{I}.$$

where R is the initial distance from the pusuer and $\Theta \in \mathcal{U}(0, \pi/2)$.

P4.1 For each $R = 1, 2, \cdots, 10$, generate $K = 100$ random $\Theta = \theta$ and execute the simple pursue strategy, and compute the average time of capture $\hat{N}$

$$\hat{N}(R) = \frac{1}{100}\sum_{k=1}^{K} N_k(R) \tag{5}$$

where $N_k(R)$ is the time of capture in the kth Monte Carlo run. Plot $\hat{N}(R)$ vs. R.

Problem 5: Multi-target pursue in a box (Option with 20 Bonus points) Devise a strategy that performs better than the one-step predictive pursue.

P5.1 Explain your strategy and provide an implementation of your strategy.

P5.2 Test your strategy and that used in Problem 4 under the setting involving three targets within a box as in Problem 3 with the following parameters:

$$\boldsymbol{v}_1[0] = \begin{bmatrix} 0.1 \\ 0.05 \end{bmatrix}, \quad \boldsymbol{R}_1[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}_1[n] = \sigma_W^2 \boldsymbol{I}.$$

$$\boldsymbol{v}_2[0] = \begin{bmatrix} -0.1 \\ 0.05 \end{bmatrix}, \quad \boldsymbol{R}_2[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}_2[n] = \sigma_W^2 \boldsymbol{I}.$$

$$\boldsymbol{v}_3[0] = \begin{bmatrix} -0.1 \\ -0.05 \end{bmatrix}, \quad \boldsymbol{R}_3[n] = 0.01\boldsymbol{I}, \quad \boldsymbol{Q}_3[n] = \sigma_W^2 \boldsymbol{I}.$$

For $\sigma_W^2 = 0, 0.02, 0.04, 0.06, 0.08, 0.1$, generate 100 random scenarios with uniformly distributed initial positions $\boldsymbol{z}_1[0], \boldsymbol{z}_2[0], \boldsymbol{a}_3[0]$ of the three targets. Let $\hat{N}^*(\sigma_W)$ be the estimated time required to capture all targets using your strategy (defined similar to (5) and $\hat{N}(\sigma_W)$ for the strategy in Problem 4. Plot in the same figure $\hat{N}^*(\sigma_W)$ and $\hat{N}(\sigma_W)$ vs. σ_W^2.