Infinite-Horizon Stochastic Optimal Control

ECE 276B, Project-3, 06-14-2023

Renu Krishna Gutta
PID: A59018210
Department of ECE
UC San Diego
rgutta@ucsd.edu

I. Introduction

The objective of this project is to achieve precise trajectory tracking for a ground differential-drive robot, aiming to address various real-life applications such as autonomous vehicles and other scenarios involving pathfinding and obstacle avoidance. The problem at hand can be formulated as a model-based infinite-horizon stochastic optimal control problem. To solve this problem, we will explore and compare two distinct approaches: (a) receding-horizon certainty equivalent control (CEC) and (b) generalized policy iteration.

The receding-horizon certainty equivalent control approach involves decomposing the infinite horizon problem into a series of finite horizon optimal control problems. Each of these finite horizon problems is subsequently solved using non-linear programming techniques. By repeatedly solving these finite horizon problems while considering the evolving system state, the robot can effectively track the desired trajectory while accounting for constraints and uncertainties.

On the other hand, the generalized policy iteration method discretizes the state-space of the system and employs an iterative algorithm to determine the optimal policy. This approach iteratively evaluates and improves the policy by updating the value function and policy parameters. By discretizing the state-space, the problem is transformed into a finite-dimensional optimization problem, enabling the use of iterative algorithms to converge to the optimal policy.

II. Problem Statement

A. Discrete-Time Kinematic Model of the Differential-Drive Robot

Consider a robot whose state $\mathbf{x_t} := (\approx; \theta_t)$ at discrete-time $t \in \mathbb{N}$ consists of its position $\mathbf{p_t} \in \mathbb{R}^2$ and orientation $\theta_t \in [-\pi, \pi)$. The robot is controlled by a velocity input $\mathbf{u_t} := (v_t, \omega_t)$ consisting of linear velocity $v_t \in \mathbb{R}$ and angular velocity (yaw rate) $\omega_t \in \mathbb{R}$. The discrete-time kinematic model of the differential-drive robot obtained from Euler discretization of the continuous-time kinematics with time interval $\Delta > 0$ is:

$$\mathbf{x_{t+1}} = \begin{bmatrix} \mathbf{p_{t+1}} \\ \theta_{t+1} \end{bmatrix} = f(\mathbf{x_t}, \mathbf{u_t}, \mathbf{w_t}) = \underbrace{\begin{bmatrix} \mathbf{p_t} \\ \theta_t \end{bmatrix}}_{\mathbf{x_t}} + \underbrace{\begin{bmatrix} \Delta\cos(\theta_t) & 0 \\ \Delta\sin(\theta_t) & 0 \\ 0 & \Delta \end{bmatrix}}_{\mathbf{G(x_t)}} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u_t}} + \mathbf{w_t} \tag{1}$$

$$t = 0, 1, 2, \cdots$$

where $\mathbf{w_t} \in \mathbf{R}^3$ models the motion noise with Gaussian distribution $\mathcal{N}(\mathbf{0}, diag(\sigma)^2)$ with standard deviation $\sigma = [0.04, 0.04, 0.004] \in \mathbf{R}^3$. The motion noise is assumed to be independent across time and of the robot state $\mathbf{x_t}$. The kinematics model in 1 defines the probability density function $p_f(\mathbf{x_{t+1}}|\mathbf{x_t}, \mathbf{u_t})$ of $\mathbf{x_{t+1}}$ conditioned on $\mathbf{x_t}$ and $\mathbf{u_t}$ as the density of a Gaussian distribution with mean $\mathbf{x_t} + \mathbf{G}(\mathbf{x_t})\mathbf{u_t}$ and covariance $diag(\sigma)^2$. The control input $\mathbf{u_t}$ of the vehicle is limited to an allowable set of linear and angular velocities $\mathcal{U} := [0, 1] \times [-1, 1]$.

B. Error State and Equations of Motion

It will be convenient to define an error state $\mathbf{e_t} := (\tilde{\mathbf{p}}_\mathbf{t}, \tilde{\theta}_t)$, where $\tilde{\mathbf{p}}_\mathbf{t} := \mathbf{p_t} - \mathbf{r_t}$ and $\tilde{\theta}_t := \theta_\mathbf{t} - \alpha_\mathbf{t}$ measure the position and orientation deviation from the reference trajectory, respectively. The equations of motion of the error state are:

$$\begin{aligned} \mathbf{e_{t+1}} &= \begin{bmatrix} \tilde{\mathbf{p}}_\mathbf{t+1} \\ \tilde{\theta}_{t+1} \end{bmatrix} = g(t, \mathbf{e_t}, \mathbf{u_t}, \mathbf{w_t}) \\ &= \underbrace{\begin{bmatrix} \tilde{\mathbf{p}}_\mathbf{t} \\ \tilde{\theta}_t \end{bmatrix}}_{\mathbf{e_t}} + \underbrace{\begin{bmatrix} \Delta\cos(\tilde{\theta}_t + \alpha_t) & 0 \\ \Delta\sin(\tilde{\theta}_t + \alpha_t) & 0 \\ 0 & \Delta \end{bmatrix}}_{\mathbf{G(e_t)}} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u_t}} + \begin{bmatrix} \mathbf{r_t} - \mathbf{r_{t+1}} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \mathbf{w_t} \end{aligned} \tag{2}$$

C. Objective

The objective is to design a control policy for the differential-drive robot to track a desired reference position trajectory $\mathbf{r_t} \in \mathbb{R}^2$ and orientation trajectory $\alpha_t \in [-\pi, \pi)$ while avoiding collisions with obstacles in the environment. There are two circular obstacles $\mathcal{C}_1$ centered at $(-2, -2)$ with radius 0.5 and $\mathcal{C}_2$ centered at $(1, 2)$ with radius 0.5. Let $\mathcal{F} := [-3, 3]^2 \backslash (\mathcal{C}_1 \cup \mathcal{C}_2)$ denote the free space in the environment. The environment, the obstacles, and the reference trajectory are illustrated in Fig. 1.

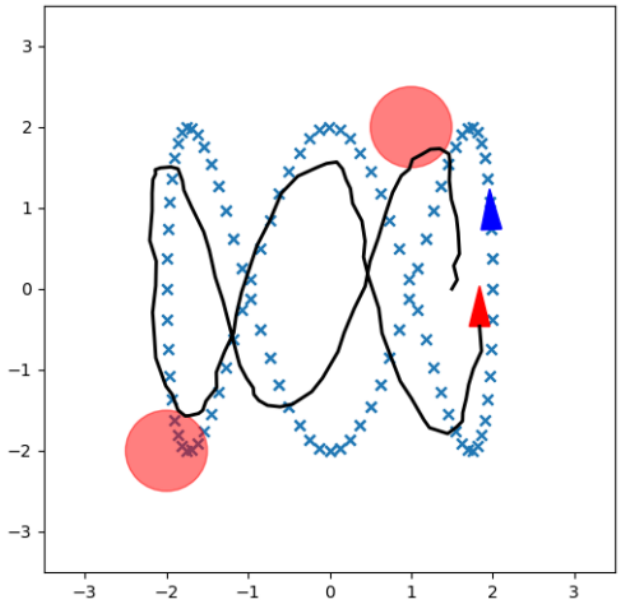

Fig. 1: Environment with baseline trajectory

We formulate the trajectory tracking with initial time τ and initial tracking error $\mathbf{e}$ as a discounted infinite-horizon stochastic optimal control problem:

$$V^*(\tau, \mathbf{e}) = \min_{\pi} \mathbb{E}\left[\sum_{t=\tau}^{\infty} \gamma^{t-\tau}\left(\tilde{\mathbf{p}}_\mathbf{t}^\mathbf{T}\mathbf{Q}\tilde{\mathbf{p}}_\mathbf{t} + q(1-\cos(\tilde{\theta}_t))^2 + \mathbf{u}_\mathbf{t}^\mathbf{T}\mathbf{R}\mathbf{u}_\mathbf{t}\right)|\mathbf{e}_\tau = \mathbf{e}\right] \tag{3}$$

$$\text{s.t. } \mathbf{e_{t+1}} = g(t, \mathbf{e_t}, \mathbf{u_t}, \mathbf{w_t}), \quad \mathbf{w_t} \sim \mathcal{N}(0, diag(\sigma)^2), t = \tau, \tau+1, \cdots$$
$$\mathbf{u_t} = \pi(t, \mathbf{e_t}) \in \mathcal{U}$$
$$\tilde{\mathbf{p}}_\mathbf{t} + \mathbf{r_t} \in \mathcal{F}$$

where $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ is a symmetric positive-definite matrix defining the stage cost for deviating from the reference position trajectory $\mathbf{r_t}$, $q > 0$ is a scalar defining the stage cost for deviating from the reference orientation trajectory α_t, and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ is a symmetric positive-definite matrix defining the stage cost for using excessive control effort.

III. Technical Approach

A. Receding-horizon Certainty Equivalent Control (Rh-CEC)

CEC is a suboptimal control scheme that applies, at each stage, the control that would be optimal if the noise variables $\mathbf{w_t}$ were fixed at their expected values (zero in our case). It reduces a stochastic optimal control problem to a deterministic optimal control problem, which can be solved more effectively. Receding-horizon, in addition, CEC approximates an infinite-horizon problem (3) by repeatedly solving the following discounted finite-horizon deterministic optimal control problem at each time step:

$$V^*(\tau, \mathbf{e}) \approx \min_{\pi_\tau, \cdots, \tau+T-1} \mathbf{q}(\mathbf{e}_{\tau+\mathbf{T}}) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau}\left(\tilde{\mathbf{p}}_\mathbf{t}^\mathbf{T}\mathbf{Q}\tilde{\mathbf{p}}_\mathbf{t} + q(1-\cos(\tilde{\theta}_t))^2 + \mathbf{u}_\mathbf{t}^\mathbf{T}\mathbf{R}\mathbf{u}_\mathbf{t}\right)|\mathbf{e}_\tau = \mathbf{e} \tag{4}$$

$$\text{s.t. } \mathbf{e_{t+1}} = g(t, \mathbf{e_t}, \mathbf{u_t}, \mathbf{0}), \quad t = \tau, \tau+1, \cdots, \tau+T-1$$
$$\mathbf{u_t} = \pi(t, \mathbf{e_t}) \in \mathcal{U}$$
$$\tilde{\mathbf{p}}_\mathbf{t} + \mathbf{r_t} \in \mathcal{F}$$

Non-linear Programming (NLP) formulation:

1) *Choosing terminal cost* $\mathbf{q}(\mathbf{e})$: As the problem involves decomposing an infinite-horizon problem into a finite version, it is necessary to define a terminal cost that captures the remaining cost beyond the assumed finite time. A meaningful approach is to formulate the terminal cost based on the magnitude of the state, similar to the stage cost formulation but with a higher weighting factor.

$$\mathbf{q}(\mathbf{e}) = \tilde{\mathbf{p}}_\mathbf{t}^\mathbf{T}\bar{\mathbf{Q}}\tilde{\mathbf{p}}_\mathbf{t} + \bar{q}(1-\cos(\tilde{\theta}_t))^2$$

By assigning a greater weight to the terminal cost, the objective function can effectively capture the importance of minimizing the state magnitude towards the end of the finite time horizon.

2) *Objective*:

$$\min_{\mathbf{U}} c(\mathbf{U}, \mathbf{E}) \equiv V^*(\tau, \mathbf{e}) \quad \text{(from equation 4)} \tag{5}$$

3) *Variables*: (i) error state, $\mathbf{e} = \begin{bmatrix}\tilde{\mathbf{p}} \\ \tilde{\theta}\end{bmatrix}$ and (ii) control, $\mathbf{u}$

4) *Parameters*: When selecting the parameters $\mathbf{Q}$, q, and $\mathbf{R}$, our objective is to regulate the error and ensure that the deviations $\tilde{\mathbf{p}}_\mathbf{t}$ and $\tilde{\theta}_t$ are minimized. The choice of these parameters should encourage the magnitudes or values of $\tilde{\mathbf{p}}_\mathbf{t}$ and $\tilde{\theta}_t$ to approach zero. The same applies for the terminal cost parameters, $\bar{\mathbf{Q}}$, $\bar{q}$. Similarly, the parameter $\mathbf{R}$ influences the control inputs $\mathbf{u_t}$ and helps balance the trade-off between achieving accurate trajectory tracking and minimizing control efforts.

5) *Time horizon*: We can vary T as a parameter. We can start with a low value, like $T = 5$ and then gradually increase to check the performance.
Time variable $t = 0, 1, 2, \cdots, T$

6) *Constraints*:
Note that we are provided with reference trajectory, i.e., we have access to $\begin{bmatrix}\mathbf{r_t} \\ \alpha_t\end{bmatrix}$, $\forall t$

$$\mathbf{e_0} = \mathbf{x_0} - \mathbf{r_0} \tag{i}$$

$$\mathbf{e_{t+1}} = \begin{bmatrix}\tilde{\mathbf{p}}_\mathbf{t+1} \\ \tilde{\theta}_{t+1}\end{bmatrix} = g(t, \mathbf{e_t}, \mathbf{u_t}, \mathbf{0}), \quad \text{with,} \tag{ii}$$

$(\tilde{\theta}_{t+1} + \alpha_{t+1})$ wrapped into $[-\pi, \pi)$ range

$$\mathbf{U_{lb}} \le \mathbf{u_t} \le \mathbf{U_{ub}}, \quad \mathbf{U_{lb}} = [0, -1], \quad \mathbf{U_{ub}} = [1, 1], \quad \forall t \tag{iii}$$

$$\mathbf{P_{lb}} \le \tilde{\mathbf{p}}_\mathbf{t} \le \mathbf{P_{ub}}, \quad \mathbf{P_{lb}} = [-3, -3], \quad \mathbf{U_{ub}} = [3, 3], \quad \forall t \tag{iv}$$

Collision avoidance; (radius $= 0.5$) (v)

$$\text{EuclideanDistance}[(\tilde{\mathbf{p}}_\mathbf{t} + \mathbf{r_t}), \mathcal{C}_1] > \text{radius} \quad \forall t$$
$$\text{EuclideanDistance}[(\tilde{\mathbf{p}}_\mathbf{t} + \mathbf{r_t}), \mathcal{C}_2] > \text{radius} \quad \forall t$$

7) *Implementation*: Once a control sequence $\mathbf{u}_\tau, \cdots, \mathbf{u}_{\tau+\mathbf{T}-\mathbf{1}}$ is obtained, CEC applies the first control $\mathbf{u}_\tau$ to the system, obtains the new error state $\mathbf{e}_{\tau+\mathbf{1}}$ at time $\tau+1$, and repeats the optimization in 4 online to determine the control input $\mathbf{u}_{\tau+\mathbf{1}}$. The total simulation time is 120sec with CEC optimization done every 0.5sec.

B. Generalized Policy Iteration (GPI)

Generalized Policy Iteration (GPI) requires state and control spaces to be finite. Hence, we are discretizing both the spaces.

1) *State and control spaces discretization*:
   - We consider state as $\mathbf{x} = [t, \delta x, \delta y, \tilde{\theta}]^T$, i.e., time instance, XY position error and the yaw angle error of the robot. And we know control, $u = [v, \omega]^T$.
   - We discretize the state and control spaces into $(n_t, n_x, n_y, n_\theta)$ and (n_v, n_ω) number of grid points, respectively.
   - $n_t = 100$ since the provided reference trajectory is periodic with period of 100.
   - The position error $\tilde{\mathbf{p}} := (\delta x, \delta y)$, yaw angle error $\tilde{\theta}$ and control input $\mathbf{u}$ are discretized over the regions $[-3, 3]^2$, $[0, 1] \times [-1, 1]$ and $[-\pi, \pi)$ respectively.
   - Wrap around is enforced on grid points of $\tilde{\theta}$, i.e., first and last grid points are adjacent.
2) *Transition probabilities*:
   - Iterate over t and the grid points of $\mathbf{e}$ and $\mathbf{u}$. For each $(t, \mathbf{e}, \mathbf{u})$, the mean of next error state will be $g(t, \mathbf{e}, \mathbf{u}, \mathbf{0})$.
   - Select the neighboring grid points within a distance of $3 \times$ std.dev around the mean error state. This determines a region around the mean where the Gaussian weights will be evaluated.
   - Use the covariance matrix $diag(\sigma)^2$, where $\sigma = [0.04, 0.04, 0.004]$, to compute the Gaussian weights for each of the neighboring grid points. The covariance matrix specifies the spread or variability of the error states.
   - Normalize the computed Gaussian weights so that they sum to 1. These are the transition probabilities.
3) *Collision avoidance*:
   - Penalize the collisions. For a state $(t, \mathbf{e})$ and control $\mathbf{u}$, check if the point $(\mathbf{e} + ref(t))$ is inside any of the two circular obstacles. If yes, make the stage cost $\ell((t, \mathbf{e}), \mathbf{u}) = \infty$.
   - Here the reference position, $ref(t) := (\mathbf{r_t}, \alpha_t)$ is known to us for all t.
4) *Stage cost*: For the non-colliding states,

$$\ell((t, \mathbf{e}), \mathbf{u}) = \tilde{\mathbf{p}_\mathbf{t}}^T Q \tilde{\mathbf{p}_\mathbf{t}} + q(1 - \cos(\tilde{\theta}_t))^2$$

   where $\mathbf{e} = (\tilde{\mathbf{p}}, \tilde{\theta})$. Q and q are design parameters.
5) *Scaling GPI to large state space*: Current implementation follows naive discretization of both state and control spaces.

GPI implementation:
Initialize V_0

$$H[(t, \mathbf{e}), \mathbf{u}, V(.)] := \ell_t(\mathbf{e}, \mathbf{u}) + \gamma \sum_{\mathbf{e}'} T_t(\mathbf{e}, \mathbf{e}', \mathbf{u}) V(\mathbf{e}')$$

State, $\mathbf{x} := (t, \mathbf{e}), \quad \gamma = 0.99$

where $T_t(\mathbf{e}, \mathbf{e}', \mathbf{u})$ is the transition probability from state $\mathbf{e}$ to $\mathbf{e}'$ at time t under control $\mathbf{u}$.

- *Policy improvement*: Iterate over each state-control pair $((t, \mathbf{e}), \mathbf{u})$ in the discretized state and control spaces. For each $\mathbf{x} = (t, \mathbf{e})$, we find

$$\pi^*(\mathbf{x}) = \arg\min_{\mathbf{u}} H[\mathbf{x}, \mathbf{u}, V(.)]$$

- *Policy evaluation*: User input - n
  Iterate over each state $\mathbf{x} = (t, \mathbf{e})$ in the discretized space. With the policy derived from the policy improvement step, update the vallue function by running n number of value evaluation iterations.

for $k = 0, 1, \cdots, n-1$ **do**
$V_{k+1}(\mathbf{x}) = H[\mathbf{x}, \pi^*(\mathbf{x}), V_k(.)]$

Algorithm 1 Generalized Policy Iteration (GPI)

1: Define $H[\mathbf{x}, \mathbf{u}, V_k(.)] := \ell(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}_{\mathbf{x}' \sim p_f(.|\mathbf{x}, \mathbf{u})}[V(\mathbf{x}')]$
2: Initialize V_0
3: **for** $k = 0, 1, 2, \cdots$ **do**
4: $\quad \pi_{k+1}(\mathbf{x}) = \arg\min_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} H[\mathbf{x}, \mathbf{u}, V_k(.)]$
5: $\quad V_{k+1} = \mathcal{B}^n_{\pi_{k+1}}[V_k], \quad$ for $n \geq 1$
6: **end for**

IV. Results

A. Rh-CEC

Reiterating the parameters, we have the following set of parameters:
Stage cost parameters: Q, q, R
Terminal cost parameters: $\bar{Q}$, $\bar{q}$
Time horizon: $T = 9$ for all simulations
Simulation duration: 120s
Time_step_size: 0.5s
Number of iterations = 240

1) Baseline trajectory tracking: Error = 2180.5, Time = 0.03s
2) $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $q = 3$, $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\bar{Q} = \begin{bmatrix} 10 & 2 \\ 2 & 10 \end{bmatrix}$, $\bar{q} = 2$

   Fig. 2: Error = 460.7, Time = 28.47s

3) $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\bar{Q} = \begin{bmatrix} 10 & 2 \\ 2 & 10 \end{bmatrix}$, $\bar{q} = 5$

$Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$, $q = 2$ for the first 100 iterations

$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $q = 2$ for the remaining iterations

Fig. 3: Error = 220, Time = 26.82s

4) $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $\bar{Q} = \begin{bmatrix} 10 & 2 \\ 2 & 10 \end{bmatrix}$, $\bar{q} = 5$

$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $q = 2$ for the first 30 iterations

$Q = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $q = 2$ for the remaining iterations

Fig. 4: Error = 174, Time = 20.81s

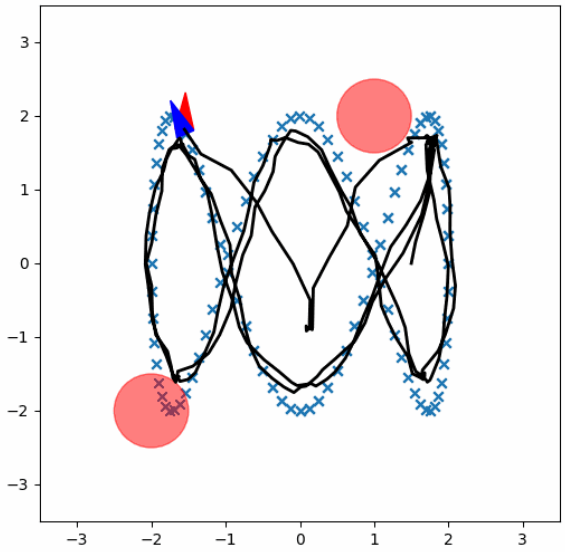

Fig. 2: Rh-CEC Error: 460.7

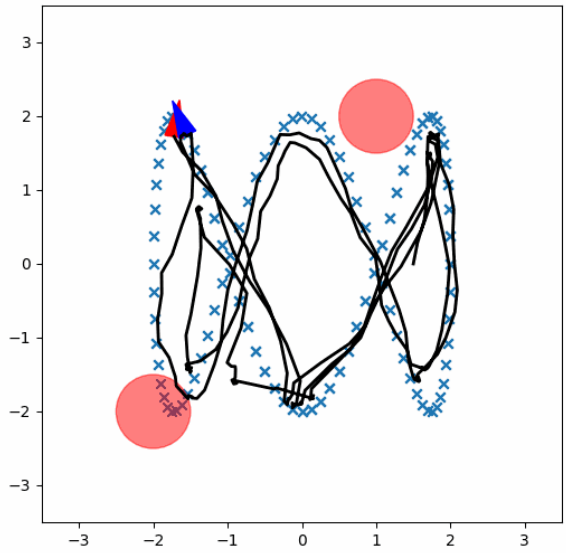

Fig. 3: Rh-CEC Error: 220

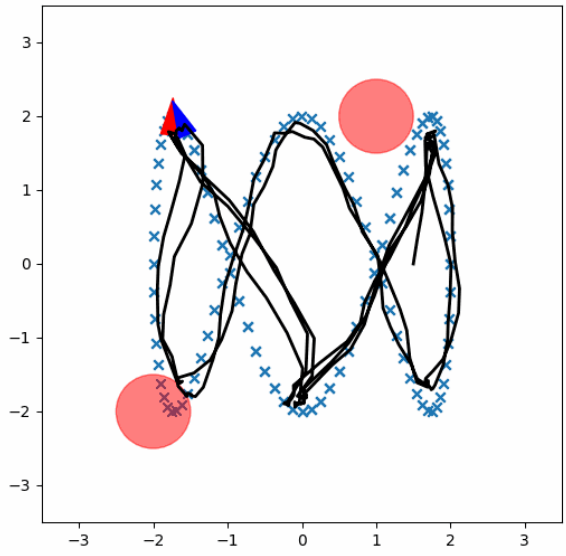

Fig. 4: Rh-CEC Error: 174

B. GPI

I implemented Generalized Policy Iteration with a naive discretization of the state and control spaces. The running time was too high and I could not produce any results.

C. Discussion

- By ignoring noise in the Rh-CEC, the uncertainties and fluctuations in the motion model are underestimated. This leads to a consistent phenomenon observed in Rh-CEC simulations where the test robot and reference robot turn in opposite directions at bends. The deterministic nature of the formulation, coupled with the absence of noise, results in the non-linear programming consistently providing similar solutions that contribute to this phenomenon.
- Upon closer observation, it is evident that initially, the reference robot is positioned to the right of the test robot. Due to the cost function formulation, the reference robot naturally tends to pull the test robot towards it. However, in order to perform a proper left turn similar to the reference robot, the test robot should move towards the left. In the absence of noise consideration, the solver consistently guides the test robot to move towards the

right, resulting in unexpected and unnatural turns to trace the reference robot's path.

- The error calculation in this context employs the L2 norm between the reference and test robot states. Consequently, more weightage is given to the position rather than the orientation. As a result, simulations may exhibit relatively low error values, but the turning behavior of the two robots at bends is completely opposite. For instance, in a simulation where the error is minimized to 174, the test and reference robots consistently turn in opposite directions at bends.
- Regarding the Generalized Policy Iteration (GPI) approach, the iteration through all state-space and control-space points proves to be time-consuming. GPI is expected to yield improved results as it incorporates the effect of noise. An adaptive approach can be explored by employing finer grids near the reference trajectory and coarser grids as one moves away from it.
- When selecting the resolution for discretization, it is crucial to consider the noise standard deviations. Insufficient resolution may result in probability weights vanishing for neighboring grids, leading to inadequate representation of the system dynamics and transition probabilities. Therefore, careful consideration should be given to ensure an appropriate resolution that accounts for the impact of noise in the probabilistic calculations.

V. Acknowledgment

I would like to thank Prof. Nikolay Atanasov and Zhirui Dai for guiding me with this project.

References

[1] https://pillow.readthedocs.io/en/stable/
[2] natanaso.github.io/ece276b/