Infinite-Horizon Stochastic Optimal Control

Rishabh Bhattacharya
Department of Mechanical & Aerospace Engineering
University of California, San Diego
ribhattacharya@ucsd.edu

***Index Terms*—infinite-horizon discounted stochastic optimal control problem, discounted finite-horizon deterministic optimal control problem, receding-horizon certainty equivalent control, CEC, generalized policy iteration, GPI**

I. Introduction

The previous two projects have focused on path-planning for robots. However, once a path is planned/computed, the robot also needs to traverse on this computed path to complete any and every task assigned to it. Optimal control/trajectory tracking is another aspect of robot motion which is critical to its function and should therefore be thoroughly investigated for effective implementation in real world scenarios.

Robot control is a particularly challenging task since all robot motions have inherent noise (either sensor or motion noise) which needs to be accounted for in order to execute motions reliably and with a degree of safety guarantees. Also, robots are physical objects which have inherent dynamics and therefore inertia. It is thus extremely important to consider the system dynamics since that might limit the robot's ability to track a reference trajectory (for eg. turning radius limitations). Usually such considerations are accounted for during the planning stage, but the robot system itself can be changing (robots carrying weights), in which case an online motion planner mandates the requirement of an online control as well.

Given such considerations, we are often tasked with finding an *optimal control policy* for a given environment, agent and task. Optimal control deals with computing a policy that minimizes an objective function over a predefined time horizon. The objective function can be specific to the task at hand, and may penalize poor performance or high control effort, behaviour that is usually undesired. The control obtained is thus *optimal* w.r.t. the objective function. This means there can be different optimal policies for the same system, but the policies will be optimal for different objectives. Oftentimes we use a value function, which represents the long term cost of following a policy. Minimizing this value function lets us compute the optimal policy.

In this project, we explore an Infinite horizon stochastic optimal control problem.

$$\begin{aligned}\text{Infinite horizon} &\rightarrow \text{no predetermined planning horizon}\\ \text{stochastic} &\rightarrow \text{probabilistic set of policies}\\ \text{optimal control} &\rightarrow \text{control minimizes a predefined objective}\end{aligned}$$

II. Problem formulation

Given: Consider a robot whose state $\mathbf{x}_t := (\mathbf{p}_t, \theta_t)$ 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:

$$\begin{aligned}\mathbf{x}_{t+1} = \begin{bmatrix}\mathbf{p}_{t+1}\\ \theta_{t+1}\end{bmatrix} &= f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)\\ &= \underbrace{\begin{bmatrix}\mathbf{p}_t\\ \theta_t\end{bmatrix}}_{\mathbf{x}_t} + \underbrace{\begin{bmatrix}\Delta\cos(\theta_t) & 0\\ \Delta\sin(\theta_t) & 0\\ 0 & \Delta\end{bmatrix}}_{\mathbf{G}(\mathbf{x}_t)} \underbrace{\begin{bmatrix}v_t\\ \omega_t\end{bmatrix}}_{\mathbf{u}_t} + \mathbf{w}_t \qquad (1)\end{aligned}$$

where $t = 0, 1, 2, \dots$ and $\mathbf{w}_t \in \mathbb{R}^3$ models the motion noise with Gaussian distribution $\mathcal{N}\left(\mathbf{0}, \operatorname{diag}(\boldsymbol{\sigma})^2\right)$ with standard deviation $\boldsymbol{\sigma} = [0.04, 0.04, 0.004] \in \mathbb{R}^3$. The motion noise is assumed to be independent across time and of the robot state $\mathbf{x}_t$. The kinematics model in Eqn. 1 defines the probability density function $p_f(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$ 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 $\operatorname{diag}(\boldsymbol{\sigma})^2$. The control input $\mathbf{u}_t$ of the vehicle is limited to an allowable set of linear and angular velocities $\mathcal{U} := [0, 1] \times [-1, 1]$.

Objective: The objective is to design a control policy for the differential-drive robot in order 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 $\mathcal{F} := [-3, 3]^2 \setminus (\mathcal{C}_1 \cup \mathcal{C}_2)$ denote the free space in the environment. The environment, the obstacles, and the reference trajectory are illustrated in Fig. 1.

Formulation: It will be convenient to define an error state $\mathbf{e}_t := \left(\tilde{\mathbf{p}}_t, \tilde{\theta}_t\right)$, where $\tilde{\mathbf{p}}_t := \mathbf{p}_t - \mathbf{r}_t$ and $\tilde{\theta}_t := \theta_t - \alpha_t$ measure the

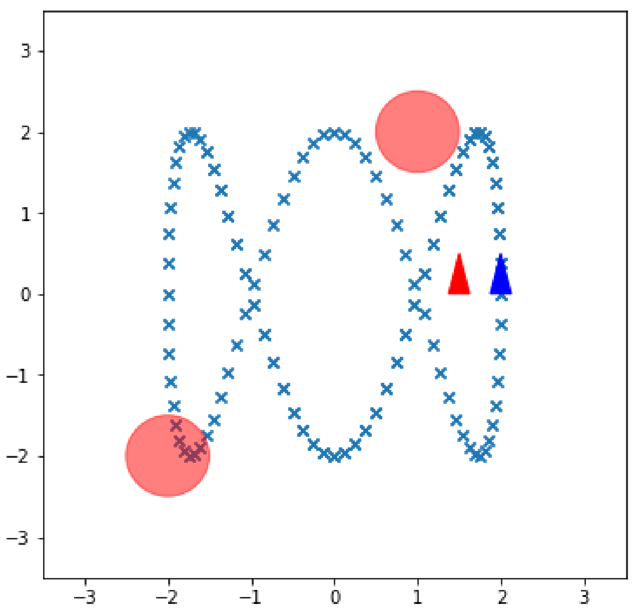

Fig. 1: Environment with obstacles (red circles), reference trajectory (cyan cross) and agent (red triangle). Reference agent is blue triangle.

position and orientation deviation from the reference trajectory, respectively. The equations of motion for the error dynamics are:

$$\mathbf{e}_{t+1} = \begin{bmatrix} \tilde{\mathbf{p}}_{t+1} \\ \tilde{\theta}_{t+1} \end{bmatrix} = g\left(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t\right)$$
$$= \underbrace{\begin{bmatrix} \tilde{\mathbf{p}}_t \\ \tilde{\theta}_t \end{bmatrix}}_{\mathbf{e}_t} + \underbrace{\begin{bmatrix} \Delta\cos\left(\tilde{\theta}_t + \alpha_t\right) & 0 \\ \Delta\sin\left(\tilde{\theta}_t + \alpha_t\right) & 0 \\ 0 & \Delta \end{bmatrix}}_{\tilde{\mathbf{G}}(\mathbf{e}_t)} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u}_t} +$$
$$\begin{bmatrix} \mathbf{r}_t - \mathbf{r}_{t+1} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \mathbf{w}_t \tag{2}$$

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

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

$$\begin{aligned} \text{s.t.} \quad & \mathbf{e}_{t+1} = g\left(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t\right) \\ & \mathbf{w}_t \sim \mathcal{N}\left(\mathbf{0}, \operatorname{diag}(\boldsymbol{\sigma})^2\right) \\ & t = \tau, \tau+1, \dots \\ & \mathbf{u}_t = \pi\left(t, \mathbf{e}_t\right) \in \mathcal{U} \\ & \tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F} \end{aligned}$$

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

The problem has been formulated as a discounted infinite-horizon stochastic optimal control problem. Some salient features of this formulation are,

- The terminal cost $\mathfrak{q}\left(\mathbf{x}_T\right)$ is no longer necessary since we never reach a terminal time-step
- As $T \to \infty$, the time-invariant motion model and stage costs lead to a time-invariant optimal value function $V^*(\mathbf{x}) = \min_\pi V^\pi(\mathbf{x})$ and associated optimal policy $\pi^*(\mathbf{x}) = \arg\min_{\pi} V^\pi(\mathbf{x})$.
- As $T \to \infty$, it is sufficient to optimize over stationary value functions $V^\pi(\mathbf{x})$ and stationary policies $\pi(\mathbf{x})$.

We consider two different approaches to tackle the problem in Eqn. 3, i) receding-horizon certainty equivalent control (III-A) and (ii) generalized policy iteration (III-B).

A. Receding-horizon certainty equivalent control (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). The main attractive characteristic of CEC is that it reduces a stochastic optimal control problem to a deterministic optimal control problem, which can be solved more effectively. Receding-horizon CEC, in addition, approximates an infinite-horizon problem by repeatedly solving the following discounted finite-horizon deterministic optimal control problem at each time step:

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

$$\begin{aligned} \text{s.t.} \quad & \mathbf{e}_{t+1} = g\left(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0}\right) \\ & t = \tau, \dots, \tau+T-1 \\ & \mathbf{u}_t \in \mathcal{U} \\ & \tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F} \end{aligned}$$

where $\mathfrak{q}(\mathbf{e})$ is a suitably chosen terminal cost. The factors determining the design of this approach are,

- T is the time horizon for computing the Value function and correspondingly the control effort. It basically translates to how further is the algorithm going to *look* in order to determine the next policy.
- $\gamma \in [0,1)$ is the discount factor. A higher discount factor places higher importance on the future states (long term cost), while a smaller γ prioritizes short term costs. An optimal range for most applications is $\gamma \in [0.8, 1]$.
- $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ is a symmetric positive-definite matrix defining the stage cost for deviating from the reference position trajectory $\mathbf{r}_t$. A higher $\mathbf{Q}$ places a higher penalty on deviations from the path (position errors in (x, y)), thus prioritizing the *performance*, potentially at the cost of a higher control action/input. Since we do not have bounds on the control cost ($\mathbf{u}_t^\top \mathbf{R} \mathbf{u}_t$) in our problem statement (as long as the control effort is within the defined bounds ($\mathbf{u} \in \mathcal{U}$), we can keep $\mathbf{Q}$ high.
- $q > 0$ is a scalar defining the stage cost for deviating from the reference orientation trajectory α_t. A higher q emphasizes tracking the reference orientation more closely, which is predominantly due to the control action ω_t.
- $\mathbf{R} \in \mathbb{R}^{2\times 2}$ is a symmetric positive-definite matrix defining the stage cost for using excessive control effort. We aim to keep this lower than the other costs $\mathbf{Q}$, q since our goal is to minimize tracking error. Keeping a higher $\mathbf{R}$ would have made sense if we had limited resources or constraints on fuel, time, distance covered, etc. However in their absence, our priority is to maximize performance, even if it comes at the cost of a higher input effort.
- $\mathfrak{q}$ is the terminal cost for the end state in the value function formulation.

The receding-horizon CEC problem is now a non-linear program (NLP) of the form:

$$\begin{aligned} \min_{\mathbf{U}} \quad & c(\mathbf{U}, \mathbf{E}) \\ \text{s.t.} \quad & \mathbf{U}_{lb} \le \mathbf{U} \le \mathbf{U}_{ub} \\ & \mathbf{h}_{lb} \le \mathbf{h}(\mathbf{U}, \mathbf{E}) \le \mathbf{h}_{ub} \end{aligned} \tag{5}$$

where $\mathbf{U} := \left[\mathbf{u}_\tau^\top, \ldots, \mathbf{u}_{\tau+T-1}^\top\right]^\top$ and $\mathbf{E} := \left[\mathbf{e}_\tau^\top, \ldots, \mathbf{e}_{\tau+T}^\top\right]^\top$. An NLP program can be solved by an NLP solver, such as CasADi [1]. Once a control sequence $\mathbf{u}_\tau, \ldots, \mathbf{u}_{\tau+T-1}$ is obtained, CEC applies the first control $\mathbf{u}_\tau$ to the system, obtains the new error state $\mathbf{e}_{\tau+1}$ at time $\tau+1$, and repeats the optimization in Eqn. 4 online in order to determine the control input $\mathbf{u}_{\tau+1}$. This online re-planning is necessary because the CEC formulation does not take the effect of the motion noise into account.

The NLP formulation is given as,

Objective

$$\min_{\mathbf{u}_\tau, \ldots, \mathbf{u}_{\tau+T-1}} \mathfrak{q}\left(\mathbf{e}_{\tau+T}\right) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau} \left(\tilde{\mathbf{p}}_t^\top \mathbf{Q} \tilde{\mathbf{p}}_t + q(1 - \cos\left(\tilde{\theta}_t\right))^2 + \mathbf{u}_t^\top \mathbf{R} \mathbf{u}_t\right)$$

where all the factors have been explained previously in this section.

Optimization variables

$$\mathbf{U} = \begin{bmatrix} v_\tau & v_{\tau+1} \cdots & v_{\tau+T-1} \\ \omega_\tau & \omega_{\tau+1} \cdots & \omega_{\tau+T-1} \end{bmatrix}_{2\times T}$$

$$\mathbf{E} = \begin{bmatrix} \tilde{\mathbf{p}}_\tau & \tilde{\mathbf{p}}_{\tau+1} \cdots & \tilde{\mathbf{p}}_{\tau+T-1} \\ \tilde{\theta}_\tau & \tilde{\theta}_{\tau+1} \cdots & \tilde{\theta}_{\tau+T-1} \end{bmatrix}_{3\times T}$$

where $\mathbf{U}, \mathbf{E}$ are initialized as symbolic variables, representing the control input and error states for the next T steps. Even though our task is to obtain the policy $\mathbf{U}$ through minimization of the objective function, we will end up with computing the error states $\mathbf{E}$ since the control and states are linked via the motion model (Eqn. 2). Do note that we only require the first control $\mathbf{U}_0 = [v_\tau, \omega_\tau]^\top$ from the complete series.

Constraints Given $\tilde{\mathbf{p}}_t = \mathbf{p}_t - \mathbf{r}_t$, where $\tilde{\mathbf{p}}_t$ is the position error and $\mathbf{r}_t$ is the position of the reference trajectory. $\tilde{\theta}_t = \theta_t - \alpha_t$ is the error in orientation, where α_t is the reference orientation. $\mathcal{C}_1$ and $\mathcal{C}_2$ are circles of radius 0.5 centered at $(-2,-2)$ and $(1,2)$ respectively.

$$\begin{aligned} [v_t, \omega_t] \in \mathcal{U} = [0,1] \times [-1,1] \quad & \text{control effort constraint} \\ \tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F} = [-3,3]^2 \backslash \left(\mathcal{C}_1 \cup \mathcal{C}_2\right) \quad & \text{free space constraint} \\ \tilde{\theta}_t + \alpha_t \in [-\pi, \pi) \quad & \text{angle constraint} \\ \tilde{\mathbf{p}}_{t+1} = f(\mathbf{e}_t, \mathbf{u}_t) \quad & \text{motion model constraint} \\ \tilde{\mathbf{p}_\tau} = \mathbf{p}_\tau - \mathbf{r}_\tau \quad & \text{initial condition constraint} \\ \tilde{\theta_\tau} = \theta_\tau - \alpha_\tau \quad & \text{initial condition constraint} \end{aligned}$$

Algorithm 1 Solution approach for trajectory tracking

```
1: $\mathbf{x}_0 \leftarrow [\mathbf{p}_0, \theta_0]$
2: for $\tau \in [0, 1 \ldots N]$ do                $\triangleright$ $N \rightarrow$ simulation horizon
3:     $\mathbf{x}_{ref,\tau} \leftarrow [\mathbf{r}_\tau, \alpha_\tau]$                $\triangleright$ reference state at time $\tau$
4:     $\mathbf{e}_\tau = \mathbf{x}_\tau - \mathbf{x}_{ref}$                $\triangleright$ error state
5:     $\mathbf{u}_\tau \leftarrow \text{CEC}(\mathbf{e}_\tau)$   $\triangleright$ determine next control from CEC based NLP solver (Alg. 2)
6:     $\mathbf{x}_{\tau+1} \leftarrow p_f(\mathbf{x}_\tau, \mathbf{u}_\tau, \mathbf{w}_\tau)$ $\triangleright$ next state according to car motion model
```

Note: As we will observe later, the noise is *pushing* our agent into the obstacle during online computations. This leads to conflicting constraints; *stay out of obstacle* and *begin planning from obstacle*. This is a drawback for the CEC algorithm, since its implementation is based on zero noise. Alternatives could be to introduce heuristics which decrease the probability of this event. We could also try to ignore the constraint if the agent is already inside the obstacle. An approach introduced in our implementation is to have a tolerance boundary ($\epsilon > 0$) around the obstacle. This places a constraint on our agent to be within $r + \epsilon$ distance away from obstacles. The value of ϵ can be reverse engineered based on a prior knowledge of the gaussian noise $\mathcal{N}(\mu, \sigma)$.

Algorithm 2 NLP solver (for CEC at the τ^{th} point in global trajectory)

```
1:  Initialize NLP parameters like $T, \mathbf{Q}, q, \mathbf{R}, \gamma$
2:  $J = 0$                                          ▷ objective function
3:  for $t \in [\tau, \tau+1, \dots \tau+T-1]$ do
4:      $\mathbf{x}_{ref,t} \leftarrow [\mathbf{r}_t, \alpha_t]$, $\mathbf{x}_t \leftarrow [\mathbf{p}_t, \theta_t]$
5:      $\mathbf{e}_t = \mathbf{x}_t - \mathbf{x}_{ref} = [\tilde{\mathbf{p}}_t \quad \tilde{\theta}_t]^\top$          ▷ error state
6:      $\tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F}$                 ▷ free space (II) constraint
7:      $\mathbf{u}_t = [v_t, \omega_t]^\top \in \mathcal{U}$              ▷ velocity (II) constraint
8:      $\tilde{\mathbf{p}}_{t+1} = f(\mathbf{e}_t, \mathbf{u}_t)$          ▷ motion model (2) constraint
9:
10:     $J \leftarrow J + \gamma^{t-\tau}\left(\tilde{\mathbf{p}}_t^\top \mathbf{Q} \tilde{\mathbf{p}}_t + q(1-\cos\left(\tilde{\theta}_t\right))^2 + \mathbf{u}_t^\top \mathbf{R} \mathbf{u}_t\right)$
                                                     ▷ sum objective symbolically
11:
12: $J \leftarrow J + \mathfrak{q}(\mathbf{e}_{\tau+T})$                    ▷ add terminal cost
13: $\mathbf{e}_\tau = \mathbf{x}_\tau - \mathbf{x}_{ref,\tau}$              ▷ initial values constraint
14: $\mathbb{U}, \mathbb{E} \leftarrow \underset{\mathbf{u}_\tau,\dots\mathbf{u}_{\tau+T-1};\mathbf{e}_\tau,\dots\mathbf{e}_{\tau+T-1}}{\arg\min} J$     ▷ run solver
15: return $\mathbb{U}_0 = \mathbf{u}_\tau$                      ▷ return initial control input
```

B. *Generalized policy iteration (GPI)*

The infinite horizon discounted optimal control problem is given by Eqn 3. We know that the optimal value function for the discounted problem satisfies

$$V^*(\mathbf{x}) = \min_{\mathbf{u}\in\mathcal{U}(\mathbf{x})}\left(\ell(\mathbf{x},\mathbf{u}) + \gamma \sum_{\mathbf{x}'\in\mathcal{X}} p_f\left(\mathbf{x}' \mid \mathbf{x},\mathbf{u}\right) V^*\left(\mathbf{x}'\right)\right) \quad \forall \mathbf{x} \in \mathcal{X} \tag{6}$$

In order to compute the value function $V^*(\mathbf{x})$ and determine the optimal policy $\pi^*(\mathbf{x})$, we can use the generalized policy iteration method. Assuming that the Value Update and Policy Improvement steps are executed an infinite number of times for all states, all combinations of the following converge,

- Any number of Value Update steps in between Policy Improvement steps
- Any number of states updated at each Value Update step
- Any number of states updated at each Policy Improvement step

We define a couple of important operators to formulate the approach more concisely,
Hamiltonian:

$$H[\mathbf{x},\mathbf{u},V(\cdot)] = \ell(\mathbf{x},\mathbf{u}) + \gamma \mathbb{E}_{\mathbf{x}'\sim p_f(\cdot\mid\mathbf{x},\mathbf{u})}\left[V\left(\mathbf{x}'\right)\right] \tag{7}$$

Policy Evaluation Backup Operator:

$$\begin{aligned}\mathcal{B}_\pi[V](\mathbf{x}) &:= H[\mathbf{x},\pi(\mathbf{x}),V] \\ &= \ell(\mathbf{x},\pi(\mathbf{x})) + \gamma \mathbb{E}_{\mathbf{x}'\sim p_f(\cdot\mid\mathbf{x},\pi(\mathbf{x}))}\left[V\left(\mathbf{x}'\right)\right]\end{aligned} \tag{8}$$

Value Iteration Backup Operator:

$$\begin{aligned}\mathcal{B}_*[V](\mathbf{x}) &:= \min_{\mathbf{u}\in\mathcal{U}(\mathbf{x})} H[\mathbf{x},\mathbf{u},V] \\ &= \min_{\mathbf{u}\in\mathcal{U}(\mathbf{x})}\left\{\ell(\mathbf{x},\mathbf{u}) + \gamma \mathbb{E}_{\mathbf{x}'\sim p_f(\cdot\mid\mathbf{x},\mathbf{u})}\left[V\left(\mathbf{x}'\right)\right]\right\}\end{aligned} \tag{9}$$

We can now state the GPI approach, which consists of three steps,

1) **Initialization**: arbitrary as long as V^π is finite
2) **Policy Improvement**: choose the action that minimizes the Hamiltonian:

$$\begin{aligned}\pi'(\mathbf{x}) &= \underset{\mathbf{u}\in\mathcal{U}(\mathbf{x})}{\arg\min} H\left[\mathbf{x},\mathbf{u},V^\pi(\cdot)\right] \\ &= \underset{\mathbf{u}\in\mathcal{U}(\mathbf{x})}{\arg\min}\left\{\ell(\mathbf{x},\mathbf{u}) + \gamma \mathbb{E}_{\mathbf{x}'\sim p_f(\cdot\mid\mathbf{x},\mathbf{u})}\left[V^\pi\left(\mathbf{x}'\right)\right]\right\}\end{aligned} \tag{10}$$

3) **Policy Evaluation**: Given π compute V^π

$$V^\pi = \mathbf{v} = (I-\gamma P)^{-1}\ell \quad \text{OR} \quad V_{k+1} = \mathcal{B}_\pi\left[V_k\right] \tag{11}$$

This step can be solved via a linear system or iteratively (value iteration) until $|V_{k+1} - V_k| < \epsilon$, where ϵ is the error tolerance. .

Algorithm 3 Generalized Policy Iteration (GPI)

```
1: Initialize $V_0$
2: for $k = 0, 1, \dots$ do
3:     $\pi_{k+1}(\mathbf{x}) = \underset{\mathbf{u}\in\mathcal{U}(\mathbf{x})}{\arg\min} H\left[\mathbf{x},\mathbf{u},V_k(\cdot)\right]$        ▷ Policy
   Improvement
4:     $V_{k+1} = \mathcal{B}^n_{\pi_{k+1}}\left[V_k\right]$,   for $n \geq 1$   ▷ Policy Evaluation
```

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. $\tilde{\theta}$ is an angle and wrap-around should enforced, while $n_t = 100$ since the provided reference trajectory is periodic with period of 100 . The position error $\tilde{\mathbf{p}}$ and control input $\mathbf{u}$ may be discretized over the regions $[-3,3]^2$ and $\mathcal{U}$. To create transition probabilities in the discretized MDP, for each discrete state $\mathbf{e}$ and each discrete control $\mathbf{u}$, we can choose the next grid points $\mathbf{e}'$ around the mean $g(t,\mathbf{e},\mathbf{u},0)$, evaluate the likelihood of $\mathcal{N}\left(\mathbf{0},\operatorname{diag}(\boldsymbol{\sigma})^2\right)$ at the chosen grid points, and normalize so that the outgoing transition probabilities sum to 1. We implement the map/control constraints and solve the MDP using GPI.

IV. Results

We have successfully implemented CEC for trajectory tracking of our agent. Here we present an analysis of the performance, by varying the parameters used for our implementation. An important note to keep in mind is that since we are using a numerical solver, there are many instances where the solver might not be able to find a feasible solution, even though we can infers by intuition that a solution should exist. A more thorough investigation is required in this regard.

The parameters that affect our solution are namely $T, \mathbf{Q}, \mathbf{R}, q, \gamma$, which have been described in detail in III-A. Tables I, II, III, IV and V present the variation in i) total runtime, ii) average iteration time and iii) final error. For our problem, we deduce that $N = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, \gamma = 0.87$ works

$\mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$			
	Total time (s)	Average iteration time (ms)	Final error (map units)
$T = 1$	1.74	7.20	2199.28
$T = 5$	Infeasible		
$T = 10$	8.44	35.13	427.50
$T = 20$	Infeasible		

TABLE I: Results by varying T (time horizon). We see that we have a couple of infeasible results, and thus $T = 10$ is a sweet spot for our simulation. $T = 1$ doesn't make sense as we are not solving for a value function in that case, just looking at the next state and determining our control.

$T = 10, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$			
	Total time (s)	Average iteration time (ms)	Final error (map units)
Q = 0.1 $\times\mathbf{I}$	8.29	34.51	477.35
Q = 0.3 $\times\mathbf{I}$	8.28	34.47	438.29
Q = 0.9 $\times\mathbf{I}$	Infeasible		
Q = I	Infeasible		

TABLE II: Results by varying $\mathbf{Q}$ (state error cost/performance penalty). We can see that increasing $\mathbf{Q}$ did in fact lead to an increase in the final error. Computation times were not affected. Higher values tend to be infeasible for the solver, but this validates our choice of $\mathbf{Q} = 0.7 \times \mathbf{I}$ (I).

best with regards to final error. There are other combinations of the hyper-parameters which lead to a reduced runtime or average iteration time, but they are not significant enough to outweigh the increase in error.

V. Conclusions

We have implemented a CEC controller for reference trajectory tracking of a mobile robot. Next steps in this approach would be to i) better understand the error function, ii) robustify

$T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, q = 10, \gamma = 0.98$			
	Total time (s)	Average iteration time (ms)	Final error (map units)
R = 0.1 $\times\mathbf{I}$	Infeasible		
R = 0.3 $\times\mathbf{I}$	Infeasible		
R = 5 $\times\mathbf{I}$	7.86	32.70	482.84
R = 10 $\times\mathbf{I}$	Infeasible		

TABLE III: Results by varying $\mathbf{R}$ (input/control cost). We see that our implementation is quite sensitive to the input cost. Our choice of $\mathbf{R} = \mathbf{I}$ (I) is thus valid, as it decreases the error compared to the other values.

$T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, \gamma = 0.98$			
	Total time (s)	Average iteration time (ms)	Final error (map units)
$q = 1$	7.88	32.82	427.10
$q = 5$	8.29	34.51	428.27
$q = 100$	8.74	36.38	369.21
$q = 1000$	Infeasible		

TABLE IV: Results by varying q (orientation cost). We have more feasible data points for a q parameter comparison. We observe that the error/performance increases with an increase in q. Even though it seems like $q = 100$ would be the better choice, we must also keep in mind that the costs are weighed somewhat equally. $\mathbf{\tilde{p}_t}^\top \mathbf{Q} \mathbf{\tilde{p}_t} \sim \|\mathbf{\tilde{p}_t}\|^2$ and $q(1 - cos(\tilde{\theta}_t))^2 \in [0, 4q]$, given $\mathbf{Q} = \mathbf{I}$. We want these errors to weight equally for our implementation in this project, unless we wanted to place importance on either one of them. We would weigh the deviations/error accordingly in that scenario. Also as we will see later in Fig. 9, the error calculation needs to be investigated since the trajectory is worse off compared to the other values.

$T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10$			
	Total time (s)	Average iteration time (ms)	Final error (map units)
$\gamma = 0.7$	Infeasible		
$\gamma = 0.85$	8.83	36.76	582.23
$\gamma = 0.87$	8.40	34.99	416.66
$\gamma = 0.9$	8.43	35.10	418.30
$\gamma = 0.95$	8.36	34.81	423.30
$\gamma = 0.98$	8.21	34.17	430.56

TABLE V: Results by varying γ (discount factor). This table is very interesting, as it shows a slightly decreasing, and then increasing behaviour of γ on our performance. The green highlighted row shows the optimal choice of hyper-parameters we have chosen for our implementation and plots. This choice of $\gamma = 0.87$ is high enough to prioritize future states, which might lead to a better (but still sub-optimal solution). The table shows a considerable variation of error vs γ, leading us to believe that the computations are quite sensitive to our choice of γ.

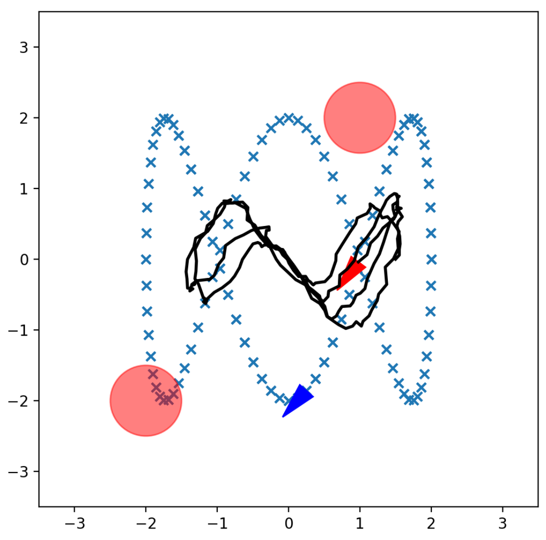

Fig. 2: $T = 1, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$. For $T = 1$, the controller is effectively looking at the next state to determine the control, which is why it is extremely inaccurate. This demonstrates the importance to have a finite/infinite horizon problem.

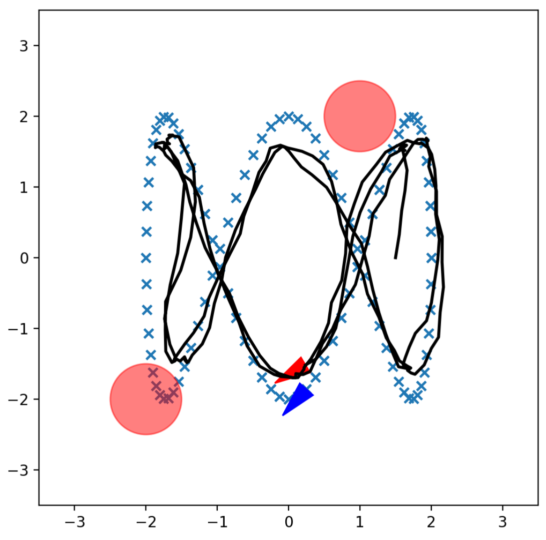

Fig. 3: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$. Compared to Fig 2, the only change is increase in T. As is evident, it has drastically improved performance.

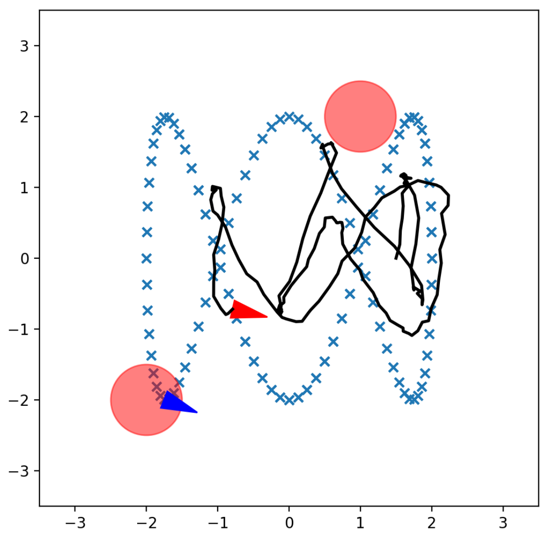

Fig. 4: $T = 10, \mathbf{Q} = 0.1 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$. With a very low performance cost $\mathbf{Q}$, the algorithm can *get away* with high errors, as they don't contribute much to the objective function. The result is a poorly performing tracking controller.

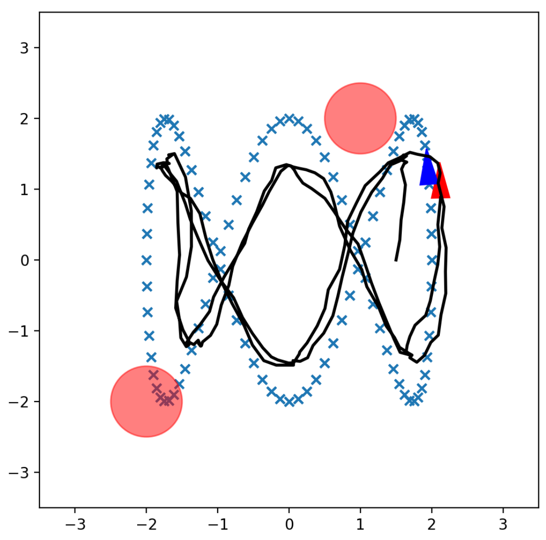

Fig. 5: $T = 10, \mathbf{Q} = 0.3 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$. Increasing the $\mathbf{Q}$ compared to Fig. 4 leads to a significant improvement in the tracking capabilities of the controller, due to a higher contribution of the error term $\tilde{\mathbf{p}}_\mathbf{t}^\top \mathbf{Q} \tilde{\mathbf{p}}_\mathbf{t}$ in the objective function.

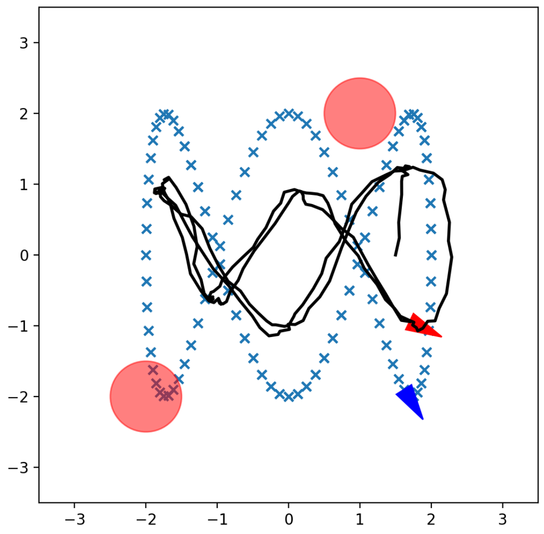

Fig. 6: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = 5 \times \mathbf{I}, q = 10, \gamma = 0.98$. A high $\mathbf{R}$ imposes a higher cost on the control effort. This essentially translates to using a very low control effort, at the expense of greater errors/deviations from the reference trajectory. Even though our formulation has bounds on the control inputs $\mathbf{u}_t \in \mathcal{U}$, there is no restriction in terms of fuel spent, time taken, resources used, etc. Thus we should try to keep a low control cost $\mathbf{R}$ for our implementation in this project.

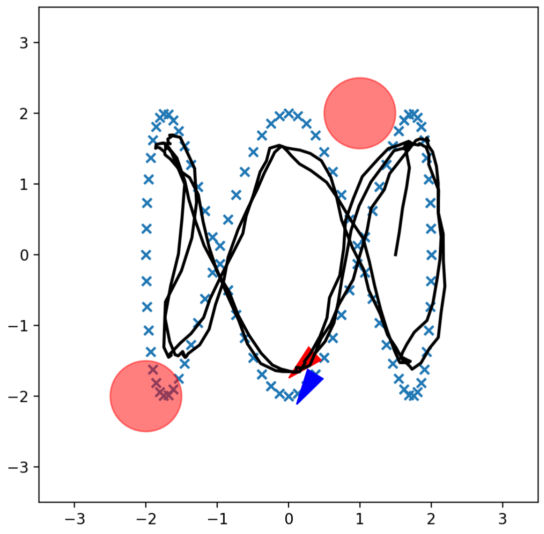

Fig. 7: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 1, \gamma = 0.98$. A low q enables a higher orientation error. Even though the trajectory might look passable, the orientation at each time step cannot be displayed accurately for just this single figure instance.

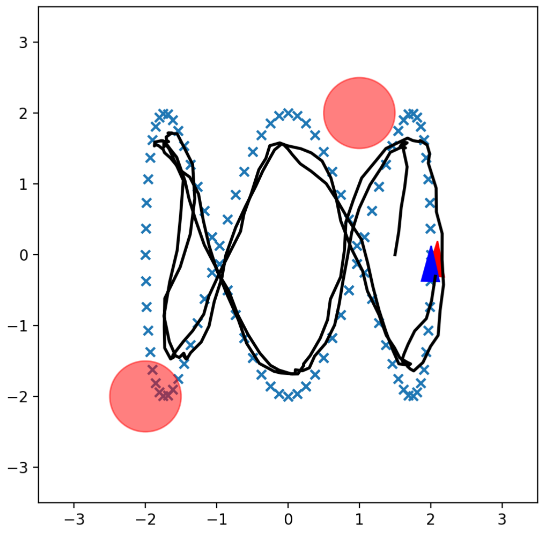

Fig. 8: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 5, \gamma = 0.98$. A higher q compared to Fig., 7 seems to have a marginally better performance, which might not be visible from this plot but rather from Table IV.

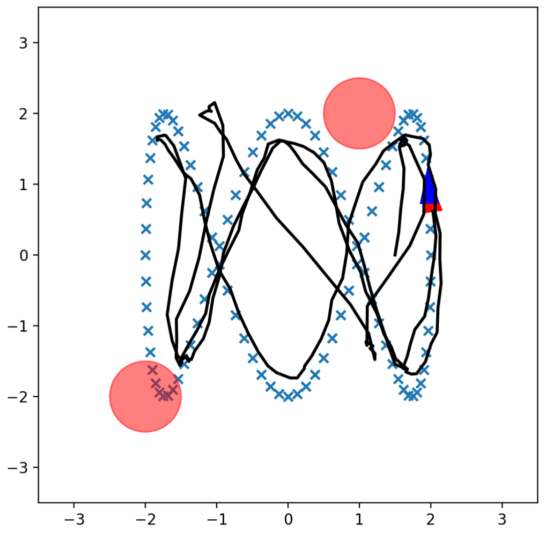

Fig. 9: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 100, \gamma = 0.98$. A very high cost on q might instinctively lead to believe lower orientation costs, but as we see, the path has deviated considerably from the reference trajectory, even though the numerical values of the error might seem lower.

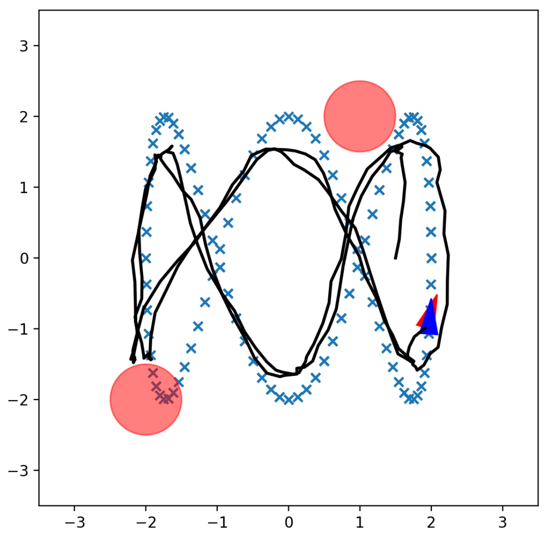

Fig. 10: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.85$. We investigate now the effect of γ on our simulation. We observe that the tracking is good enough, except for the $\mathcal{C}_1$ (obstacle at $(-2, -2)$), which will turn up again in our further analyses.

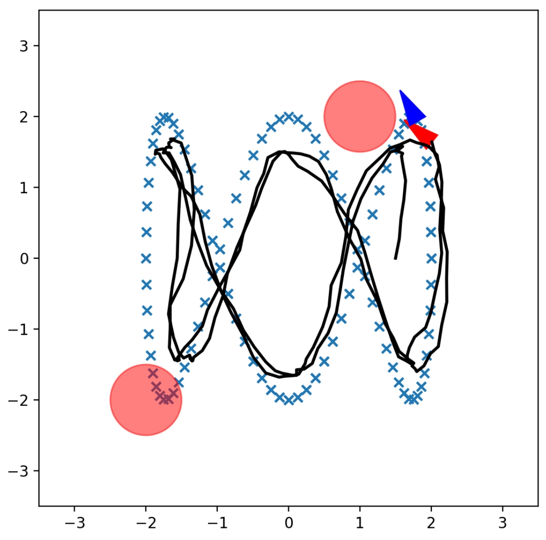

Fig. 11: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.87$. Compared to Fig. 10, a 0.02 increase in γ led to a distinctly different path tracked. We observe that the right hand part of the plot remains the same across variations in γ, but the left side changes due to the reference trajectory actually entering into the obstacle. The agent makes a loop in the upper-left portion of the environment, which was not present in Fig. 10. This might be because the agent is looking ahead $T = 10$ steps, and computes this loop to be the most *appropriate* control.

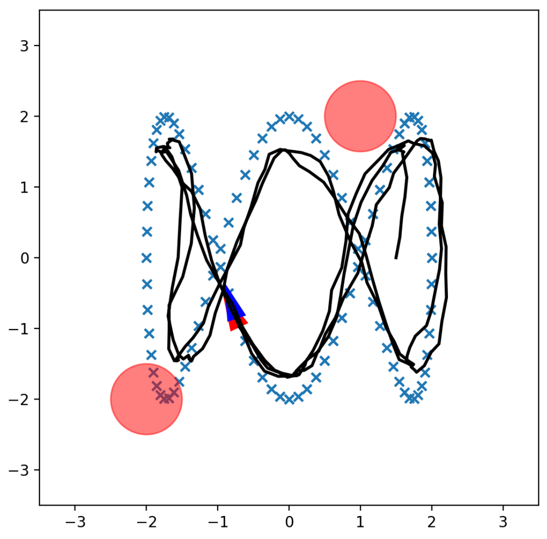

Fig. 12: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.9$. We don't find a drastic change in performance, but specifics can be looked into from Table V.

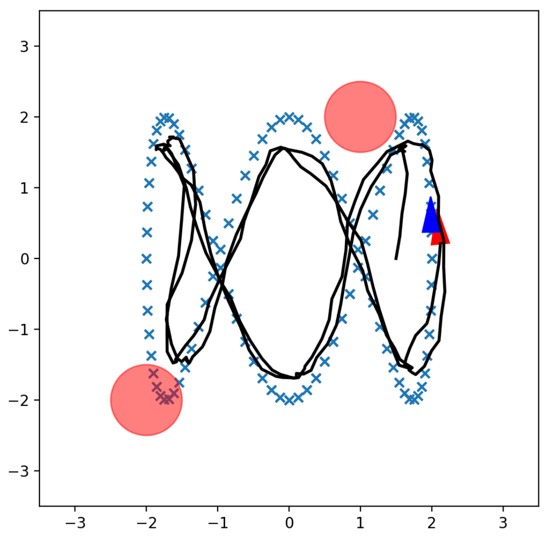

Fig. 13: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.95$. We don't find a drastic change in performance, but specifics can be looked into from Table V.

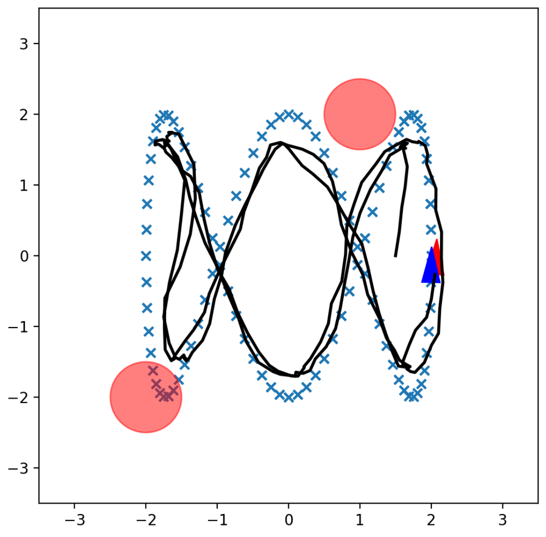

Fig. 14: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.98$. We don't find a drastic change in performance, but specifics can be looked into from Table V.

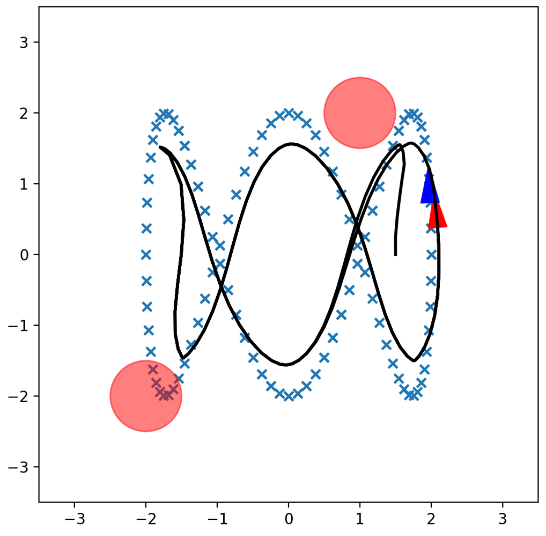

Fig. 15: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.87$. (Noiseless - no gaussian $\mathbf{w}_t$). The path tracked is smooth, due to the absence of any noise.

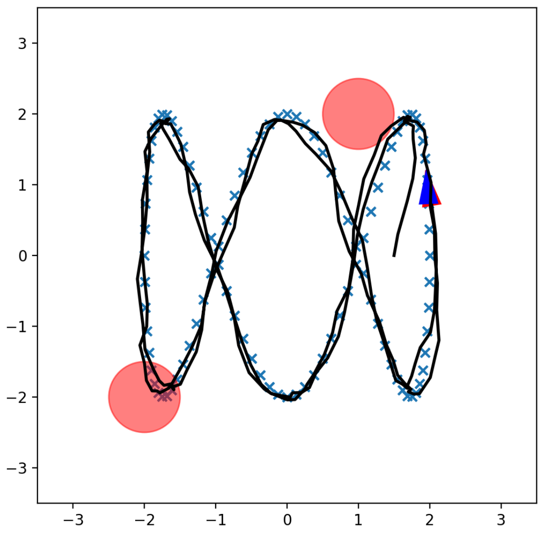

Fig. 16: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.87$. (Without obstacles, but with gaussian noise $\mathbf{w}_t$). As is evident, the path tracked is precisely what we expect, since obstacles are absent. An interesting aspect to note is that the only difference between this plot and the other plots is the absence of obstacles. This means that the obstacles are somehow causing the deviations in the previous plots, which should be investigated further.

the controller by investigating the obstacle constraints and iii) try to reason for infeasibilities in the NLP solver for minute parameter changes.

References

[1] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

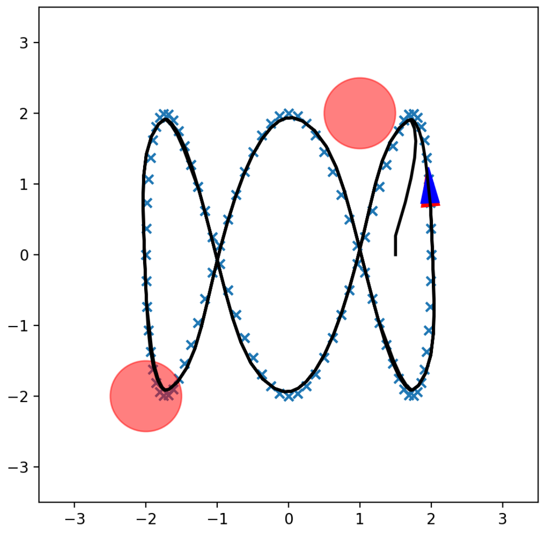

Fig. 17: $T = 10, \mathbf{Q} = 0.7 \times \mathbf{I}, \mathbf{R} = \mathbf{I}, q = 10, \gamma = 0.87$. (Without obstacles or gaussian noise $\mathbf{w}_t$). This plot shows that our implementation is accurate, and the tracker-controller is tracking the reference accurately. An improvement to this design could lead to a more robust controller, which could track with lower errors/path deviations even with obstacles and motion noise.