Trajectory Tracking

Surya Lakshmi Subba Rao Pilla
Electrical and Computer Engineering
Course: Planning and Learning in Robotics

I. Introduction

Trajectory tracking is a critical aspect of control systems that involves guiding a system or object along a desired path or trajectory. It finds extensive applications across various domains, including robotics, autonomous vehicles, aerospace, and industrial automation. The ability to accurately follow predefined trajectories is essential for achieving precise and reliable performance in these systems. Over the years, significant advancements have been made in trajectory tracking techniques, enabling enhanced control and improved performance.

In this project, the discounted infinite-horizon stochastic optimal control problem is solved using two approaches. In part 1 receding-horizon certainty equivalent control (CEC) is used to solve the problem. In part 2 Generalized Policy Iteration (GPI) is used to solve the problem.

II. Problem Formulation

Given a ground differential-drive robot, the aim is to consider a safe trajectory. The state $x_t := (p_t, \theta_t)$ at discrete-time t $\in \mathbb{N}$ consists of its position $p_t \in \mathbb{R}^2, \theta_t \in [-\pi, \pi)$. The robot is controlled by a velocity input $u_t := (v_t, \omega_t)$ 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:

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

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

Given the above details of the environment and the robot, the objective is to track given reference trajectory position

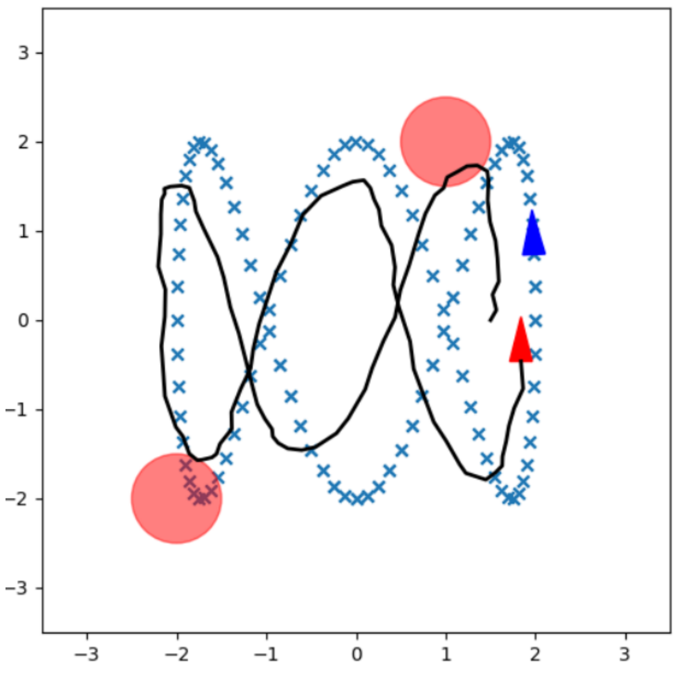

Fig. 1. Environment

$\mathbf{r}_t \in \mathbb{R}^2$. and orientation trajectory $\alpha_t \in [-\pi, \pi)$ while avoiding collisions with obstacles in the environment. There are two circular obstacles $C1$ centered at $(-2, -2)$ with radius 0.5 and $C2$ centered at $(1, 2)$ with radius 0.5. Let $F := [-3, 3]^2 \setminus (C1 \cup C2)$ denote the free space in the environment. The environment, the obstacles, and the reference trajectory are illustrated in Fig. 1.

Defining error state $e_t = (\tilde{p}_t, \tilde{\theta}_t)$, where $\tilde{p}_t = p_t - r_t$ and $\tilde{\theta}_t = \theta_t - \alpha_t$ measure the position and orientation deviation from the reference trajectory, respectively. The equations of motion of the error state are:

$$e_{t+1} = \begin{bmatrix} \tilde{p}_{t+1} \\ \tilde{\theta}_{t+1} \end{bmatrix} = g(t, e_t, u_t, w_t)$$

$$\begin{bmatrix} \tilde{p}_t \\ \tilde{\theta}_t \end{bmatrix} + \begin{bmatrix} \Delta\cos(\tilde{\theta}_t + \alpha_t) & 0 \\ \Delta\sin(\tilde{\theta}_t + \alpha_t) & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \begin{bmatrix} r_t - r_{t+1} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \mathrm{W}_t . \tag{2}$$

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

$$V^*(\tau, \mathbf{e}_t) = \min_{\pi} E \sum_{t=\tau}^{\infty} \gamma^{t-\tau} \left(\tilde{p}_t^T Q \tilde{p}_t + q(1 - \cos(\tilde{\theta}_t))^2 + \right. \\ u_t^T R u_t | e_\tau = e \quad (3)$$

subject to:
$e_{t+1} = g(t, e_t, u_t, w_t)$,
$w_t \sim N(0, \text{diag}(\sigma)^2), t \quad = \tau, \tau + 1, \ldots$
$u_t = \pi(t, e_t) \in U$,
$\tilde{p}_{t+1} + r_t \in$F

III. Technical Approach

The two methods suggested are used to solve minimization problem for solving the tracking error.

A. *Receding Horizon Certainty Equivalent Control*

The Receding Horizon Certainty Equivalent Control (CEC) is a control approach that operates at each time step by using a fixed control value, assuming the noise in the robot's motion model is fixed at the mean of the distribution, which in our case is 0. This assumption is based on considering a zero mean Gaussian noise, as explained in the preceding section. By fixing the noise, the original stochastic problem is transformed into a deterministic optimal control problem, which is somewhat simpler to solve. Moreover, rather than solving the tracking problem for an infinite time horizon, CEC solves it for a finite horizon. Receding-horizon CEC approximates an infinite-horizon problem by repeatedly solving the following discounted finite-horizon deterministic optimal control problem at each time step:

$$\begin{aligned} V^*(\tau, \mathbf{e}) = \min_{\mathbf{u}\tau, \ldots, \mathbf{u}\tau+T-1} \quad & Q(\mathbf{e}_{\tau+T}) \\ & + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau} \left(\mathbf{p}_t^T \mathbf{Q} \mathbf{p}_t \right. \\ & \left. + q(1 - \cos\theta_t)^2 + \mathbf{u}_t^T \mathbf{R} \mathbf{u}_t \right) \\ \text{s.t.} \quad & \mathbf{e}_{t+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, 0) \\ & \mathbf{u}_t \in \mathcal{U} \\ & \mathbf{p}_t + \mathbf{r}_t \in \mathcal{F} \end{aligned}$$

$V^*(\tau, \mathbf{e})$ represents the optimal value function, $\mathbf{e}$ denotes the error states, $\mathbf{u}\tau, \ldots, \mathbf{u}\tau + T - 1$ represent the control inputs.

The receding-horizon CEC problem is formulated as a non-linear program (NLP) with the following general form:

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

Where
$\mathbf{U}$ (control inputs) $= [\mathbf{u}0, \ldots, \mathbf{u}T - 1]^T$
$\mathbf{E}$ (error states) $= [\mathbf{e}0, \ldots, \mathbf{e}_T]^T$.
$c(\mathbf{U}, \mathbf{E})$ is the cost function to be minimized.
$h(\mathbf{U}, \mathbf{E})$ if the constraint functions.
$\mathbf{U}_{\text{lb}}$ and $\mathbf{U}_{\text{ub}}$ are the lower and upper bounds on $\mathbf{U} := [0, 1]X[-1, 1]$.
$\mathbf{h}_{\text{lb}}$ and $\mathbf{h}_{\text{ub}}$ are the lower and upper bounds on $h(\mathbf{U}, \mathbf{E})$.

Implementation of optimization problem in CasADi

The optimization problem is solved using CasADi library in python. The formulation of the above problem in CasADi is given below:

- The function initializes some variables and matrices, such as Q_p, q, and R_p.
- It calculates Q(2x2) and R(2x2) based on the values of Q_p, q, and R_p.
- It calculates the err_x, err_y, and err_theta based on the cur_state and ref values.
- It creates an optimization problem using the Opti class from the CasADi library.
- It defines the decision variables U (control inputs) and E (error states) using opti.variable.
- It sets the initial error value using opti.parameter.
- It sets up the objective function obj, which is the sum of a series of terms computed in a for loop.
- The loop iterates N times, where N is set to 10. Each iteration updates the objective function.
- It adds additional constraints using opti.subject_to.
- Finally, it solves the optimization problem using the IPOPT solver (opti.solver('ipopt')) and retrieves the optimal control inputs U from the solution.
- The function returns the first column of the optimal control inputs U (sol.value(U)[:,0]).
- The constraints for the control inputs $\mathbf{u}_t \in \mathcal{U}$ are defined as:
  - Linear velocity bound: $0 \leq v \leq 1$
  - Angular velocity bound: $-1 \leq \omega \leq 1$
- The constraints for the robot space coordinates $(x, y) = \tilde{p}_t + \tilde{r}_t \in \mathcal{F}$ are defined as:
  - $-3 \leq x \leq 3$
  - $-3 \leq y \leq 3$
  - $(x + 2)^2 + (y + 2)^2 > 0.5^2$
  - $(x - 1)^2 + (y - 2)^2 > 0.5^2$
  - $\mathbf{e}_{t+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, 0)$.

Finally, it solves the optimization problem using the IPOPT solver (opti.solver('ipopt')) and retrieves the optimal control inputs U from the solution.

B. *Generalized Policy Iteration*

The object of this approach is to apply Generalized Policy Iteration(GPI) to solve the infinite-horizon stochastic optimal control problem. To achieve this value iteration algorithm is applied to the equation 3. First the control space and state space have to be discretized.

- Reference trajectory: The reference trajectory consists of a periodic motion with a period of 100 time steps. Time is represented by 100 points incrementing by 0.5, ranging from 0 to 50. Time (t): 0, 0.5, 1, ..., 49, 49.5, 50.
- Adaptive grid for robot position: Create error variables ($error_x$ and $error_y$) to define an adaptive grid for the robot's position.
  $error_x$: [-3, -0.5, -0.25, -0.15, -0.05, 0.05, 0.15, 0.25, 0.5, 3]
  $error_y$: [-3, -0.5, -0.25, -0.15, -0.05, 0.05, 0.15, 0.25, 0.5, 3]
- Discretization of orientation error: Discretize the orientation error (θ) using a predefined set of values. θ: [-180°, -90°, -45°, -27°, -9°, 9°, 27°, 45°, 90°, 180°]
- State space discretization: Divide the state space into grids with dimensions $x_{steps} = y_{steps} = 9$, $time_{steps} = 100$, and $\theta_{steps} = 9$. This results in a total state space size of $|X| = 71,000$.
- Control space discretization: Discretize the control space by defining grids for linear velocity (v_t) and angular velocity (ω_t). A step size of 10 is chosen, leading to a control space cardinality of $|U| = 100$.
  Linear velocity (v_t) : [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
  Angular velocity (ω_t) : [-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1]
- Probability with noise: Employ the expectation method to calculate the probabilities of transitioning to the next state, considering the presence of noise in the model.
- GPI algorithm: Apply the Generalized Policy Iteration (GPI) algorithm to compute the value function and derive the optimal policy. This optimization aims to minimize tracking error while accounting for the influence of noise.

Considering the goal of the GPI algorithm to address the tracking error problem in the presence of noise, it is essential to account for the potential multiple states resulting from a given error and control sequence. To compute the value function of a state using the expected value function of all possible transitions, we need to determine the transition probabilities to these states. Given that the noise follows a Gaussian distribution with covariance diag(σ^2), we can construct the state transition matrix as follows: The expected next state is denoted as et+1 = g(t, et, ut, 0). Considering the noise, we may land in the vicinity of et+1. To account for this uncertainty, we select the five closest points to et+1 in the discretized state space grid. We calculate the likelihood of these points, xi for i = 1,2,3,4,5, assuming a Gaussian noise distribution with a mean μ = g(t, et, ut, 0) and covariance $\sum$ = diag(0.04, 0.04, 0.004), using the formula:

$$p(xi) = \frac{1}{\sqrt{(2\pi)^3|\Sigma|}} e^{-\frac{1}{2}(xi-\mu)^T\Sigma^{-1}(xi-\mu)}$$

- The transition probability is obtained by normalizing the likelihood of the above points, resulting in a vector P with elements P1, P2, P3, P4, and P5, calculated as:

$$P = \left[\frac{P1}{\sum_{i=1}^{5} p(xi)}, \frac{P2}{\sum_{i=1}^{5} p(xi)}, \frac{P3}{\sum_{i=1}^{5} p(xi)}, \frac{P4}{\sum_{i=1}^{5} p(xi)}, \frac{P5}{\sum_{i=1}^{5} p(xi)}\right]$$

This transition matrix serves as the motion model for the Markov Decision Process (MDP) involved in solving the tracking error problem through the Value Iteration algorithm.

The stage cost for the MDP process is defined as:

$$l(x_t, u_t) = \tilde{p}_t^T Q_p \tilde{p}_t + q(1 - \cos\tilde{\theta}_t)^2 + u_t^T R_u u_t$$

Here, $\tilde{p}_t$ represents the position error, $\tilde{\theta}_t$ denotes the orientation error, Q_p and R_u are matrices for weighting, u_t is the control input, and the penalty term accounts for potential constraints or collisions.To further enhance the effectiveness of the cost function, we introduce a penalty term based on the proximity of the position error to the reference trajectory. The penalty is determined by the following condition:
$||\tilde{p}_t + r_t||^2 < 0.5$, give penalty of 50
$||\tilde{p}_t + r_t||^2 > 0.5$, give penalty of 0

The below Value iteration algorithm is then applied using the above defined MDP to get optimal control values.

Algorithm 1 Value Iteration

```
Require: L step cost matrix, P stochastic transition matrix
1: $V_0 \leftarrow R^{|X|}$
2: for $k \leftarrow 0, 1.....\ n$ do
3:     $Q \leftarrow L + \gamma P V_k$
4:     $V_{k+1} \leftarrow \min_{u\in U} Q$
5:     $\pi \leftarrow argmin_{u\in U} Q$
6:     if $||V_{k+1}$ - $V_k||_2 \leq$ 1e -9 then
7:         return $\pi$
8:     end if
9: end for
```

IV. Results

The results of both CEC and GPI are given in this section. Effect of cost parameters (Q,R,q) are compared.

A. *Receding Time Horizon Certainty Equivalent Control*

The figures 1-6 illustrate the impact of the parameters Q, q, R, T on the trajectory tracking using CEC method. The following observations can be made:

- Impact of Q: The Q penalizes the controller for error in position tracking. Therefore, it is also evident from the Table I and figure 1 that the error decreases as Q value increases from 2 to 10.

TABLE I
CEC CONTROLLER IMPACT OF Q

S.No	Q	q	R	Time Horizon	Avg Time(ms)	Tracking Error
1	10	5	10	15	52	119.06
2	20	5	10	15	54	112.63
3	30	20	10	15	54	110.12

- Impact of q: q tracks the reference orientation. Therefore increasing q value improves the performance or decreases the error. Refer Table II.

TABLE II
CEC CONTROLLER IMPACT OF Q

S.No	Q	q	R	Time Horizon	Avg Time(ms)	Tracking Error
1	30	5	10	15	54	110.12
2	30	15	10	15	54.56	108.28
3	30	25	10	15	55.79	106.73

- Impact of R: R penalizes the controller for putting too much control effort. Therfore as we increase R the error increases. Refer Table III.

TABLE III
CEC CONTROLLER IMPACT OF R

S.No	Q	q	R	Time Horizon	Avg Time(ms)	Tracking Error
1	30	5	10	15	54	110.12
2	30	5	20	15	53	115.39
3	30	5	30	15	51	120.1

- Time horizon: CEC controller converts infinite to finite horizon optimal control. Thereofre choosing a longer horizon will produce controls that have less error. Refer Table IV.

TABLE IV
CEC CONTROLLER IMPACT OF T

S.No	Q	q	R	Time Horizon	Avg Time(ms)	Tracking Error
1	30	5	10	5	20	77.82
2	30	5	10	10	35	105.1
3	30	5	10	20	71.23	135.45

B. Generalized Policy Iteration

The current scenario involves the utilization of the GPI (Generalized Policy Iteration) controller, which aims to optimize the agent's actions based on a provided reference path. However, it has been observed that the agent's movements deviate from the expected path, following an unusual trajectory. Figure 9 illustrates the output obtained, which did not align with the desired path within the expected timeframe. The code functions encompass all the necessary equations. The code takes more than 1 hour to run and generate a single optimal path making it difficult to run multiple times.

Collaboration: *Discussed the problem with the 2 fellow students: Dhruv Talwar

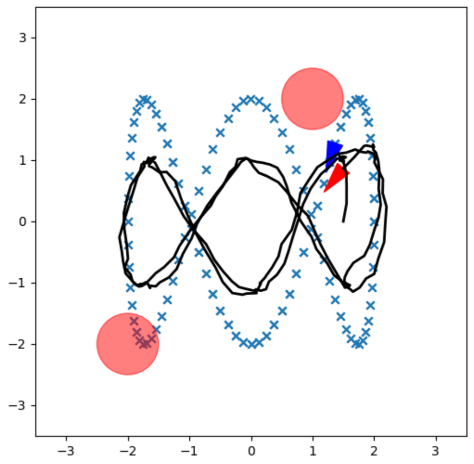
Figure 1: Q = 2, q = 5, R = 10, T = 15

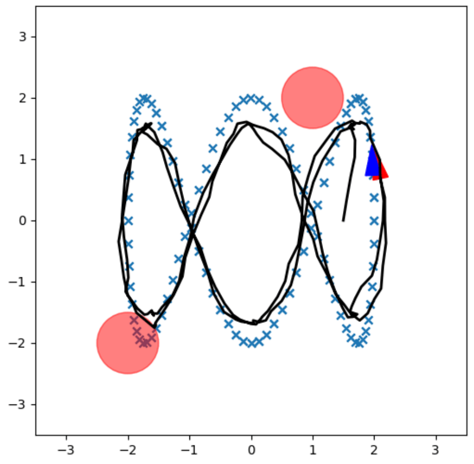
Figure 2: Q = 10, q = 5, R = 10, T = 15

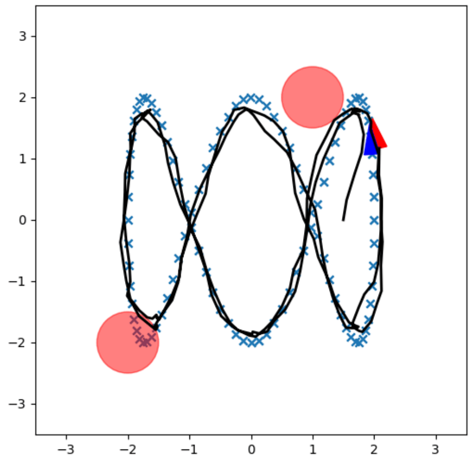
Figure 3: Q = 30, q = 5, R = 10, T = 15

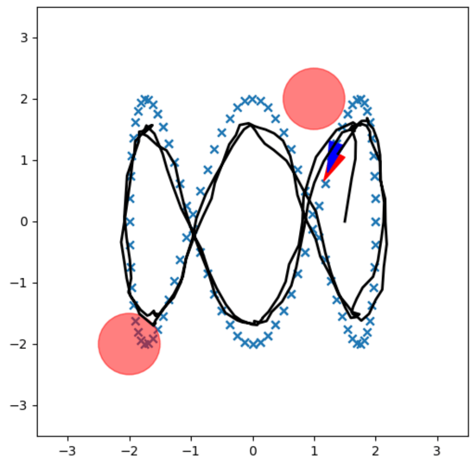

Figure 4: Q = 30, q = 5, R = 30, T = 15

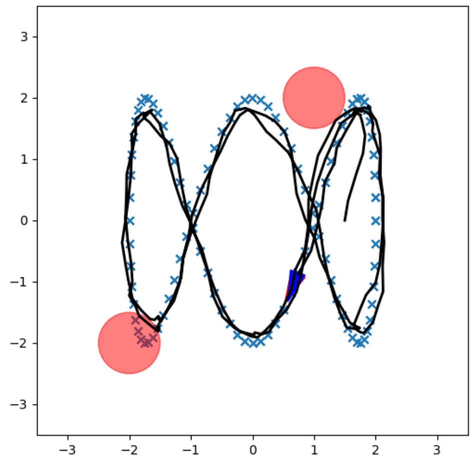

Figure 5: Q = 30, q = 25, R = 10, T = 15

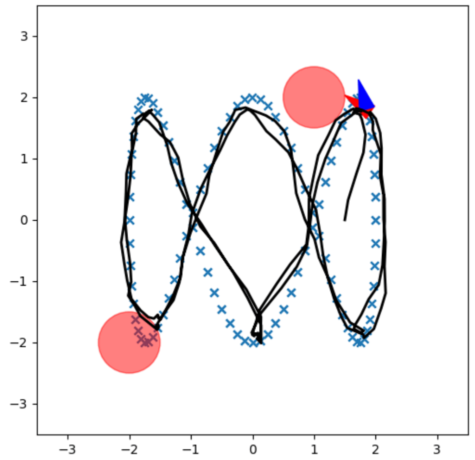

Figure 6: Q = 30, q = 5, R = 10, T = 20

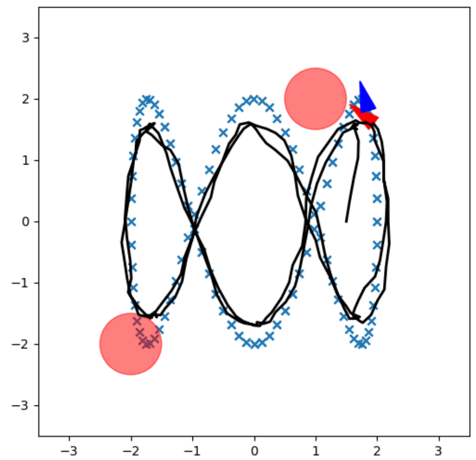

Figure 7: Q = 10, q = 20, R = 10, T = 15

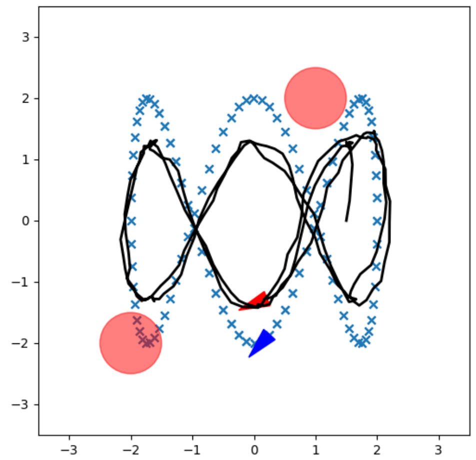

Figure 8: Q = 10, q = 20, R = 25, T = 15

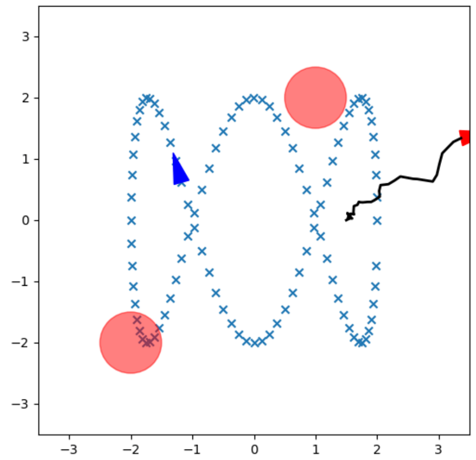

Figure 9: GPI