Infinite-Horizon Stochastic Optimal Control

Vaibhav Bishi
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, USA
vbishi@ucsd.edu

***Abstract*—This project presents an approach for safe trajectory tracking for a ground differential-drive robot. The tracking problem is formulated as a discounted infinite-horizon stochastic optimal control problem. In this project, receding-horizon certainty equivalent control (CEC) and Generalized Policy Iteration (GPI) are implemented to solve the problem. The results show that the robot successfully tracks the trajectory as traced by the reference agent while avoiding any obstacles.**

Keywords—motion control, trajectory tracking, differential-drive, infinite-horizon, stochastic optimal control, NLP, CEC, GPI

I. Introduction

Motion control and planning is one of the most crucial problems in autonomous robotics. One such typical motion control problem is *trajectory tracking*. It is concerned with designing optimal controls to enforce a robot to reach and follow a time parameterized reference. The degree of difficulty involved in solving this problem depends on the robot configuration and its motion model.

In this project, we consider safe trajectory tracking for a ground robot that follows a differential-drive model with some motion noise. One naïve way to deal with this is to implement a baseline proportional-error controller, but this would generate a large tracking error. A better approach is to formulate this tracking problem as a discounted infinite-horizon stochastic optimal control problem. The stochastic element deals with any uncertainty in the motion model. An optimal control policy can be then be designed to track any desired reference trajectory while avoiding any possible obstacles. In this project, we consider two different control schemes for solving the optimal control problem, namely, CEC and GPI, and compare their performance.

Trajectory tracking has potential applications in gaming, autonomous driving, robotic manipulators and missile technology, where the agent has to trace a reference path with minimum error while simultaneously avoiding any possible collisions.

II. Problem Statement

The objective of this project is to design an optimal control policy for a differential-drive robot for tracking a desired reference trajectory while avoiding obstacles. In this project, the trajectory tracking problem is formulated as a discounted infinite-horizon stochastic optimal control problem. The elements of the problem are described as follows.

Robot

State:

The robot in this project is a point object with its state defined as $\boldsymbol{x_t} \coloneqq (\boldsymbol{p_t}, \theta_t)$ at discrete-time $t \in \mathbb{N}$, where $p_t \in \mathbb{R}^2$ refers to its position and $\theta_t \in [-\pi, \pi)$ refers to its orientation.

Control:

The robot is controlled by a velocity input $\boldsymbol{u_t} \coloneqq (v_t, \omega_t)$ consisting of linear velocity $v_t \in \mathbb{R}$ and angular velocity (yaw rate) $\omega_t \in \mathbb{R}$. The control input $\boldsymbol{u_t}$ is limited to an allowable set of linear and angular velocities $\mathcal{U} \coloneqq [0,1] \times [-1,1]$.

Motion model:

The discrete-time kinematic motion model of the differential-drive robot obtained from Euler discretization of the continuous-time kinematics with time interval $\Delta > 0$ is defined as

$$\boldsymbol{x_{t+1}} = \begin{bmatrix} \boldsymbol{p_{t+1}} \\ \theta_{t+1} \end{bmatrix} = f(\boldsymbol{x_t}, \boldsymbol{u_t}, \boldsymbol{w_t})$$

$$= \begin{bmatrix} \boldsymbol{p_t} \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta\cos(\theta_t) & 0 \\ \Delta\sin(\theta_t) & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \boldsymbol{w_t}$$

where, $\boldsymbol{w_t} \in \mathbb{R}^3$ models the motion noise with Gaussian distribution $\mathcal{N}(\boldsymbol{0}, diag(\boldsymbol{\sigma})^2)$ 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 robot state $\boldsymbol{x_t}$.

Reference trajectory

The desired reference trajectory that the robot has to track is constituent of position trajectory $\boldsymbol{r_t} \in \mathbb{R}^2$ and orientation trajectory $\alpha_t \in [-\pi, \pi)$.

Configuration space

The free space in the environment is denoted by $\mathcal{F} \coloneqq [-3,3]^2 \backslash (\mathcal{C}_1 \cup \mathcal{C}_2)$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ refer to circular obstacles centered at $(-2,-2)$ and $(1,2)$ each with radius 0.5.

Error state

The error state is defined as $\boldsymbol{e_t} \coloneqq (\tilde{\boldsymbol{p}}_t, \tilde{\theta}_t)$, where $\tilde{\boldsymbol{p}}_t \coloneqq \boldsymbol{p_t} - \boldsymbol{r_t}$ and $\tilde{\theta}_t \coloneqq \theta_t - \alpha_t$ measure the position and orientation deviation from the reference trajectory respectively. The equations of motion for the error dynamics are defined as

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

$$= \begin{bmatrix} \tilde{\boldsymbol{p}}_t \\ \tilde{\theta}_t \end{bmatrix} + \begin{bmatrix} \Delta\cos(\tilde{\theta}_t + \alpha_t) & 0 \\ \Delta\sin(\tilde{\theta}_t + \alpha_t) & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} + \begin{bmatrix} \boldsymbol{r_t} - \boldsymbol{r_{t+1}} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \boldsymbol{w_t}$$

The trajectory tracking is formulated as a discounted infinite-horizon stochastic optimal control problem with initial time τ and initial tracking error $\boldsymbol{e}$ as follows.

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

s.t.

$$\boldsymbol{e}_{t+1} = g(t, \boldsymbol{e}_t, \boldsymbol{u}_t, \boldsymbol{w}_t)$$
$$\boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, diag(\boldsymbol{\sigma})^2)$$
$$\boldsymbol{u}_t = \pi(t, \boldsymbol{e}_t) \in \mathcal{U}$$
$$\tilde{\boldsymbol{p}}_t + \boldsymbol{r}_t \in \mathcal{F}$$

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

Now, the objective is to obtain the optimal control policy by solving the minimization problem subject to given constraints for the robot to track the reference trajectory.

III. Technical Approach

In this project, we implement two different approaches for solving the optimal control problem for trajectory tracking, namely, receding-horizon certainty equivalent control (CEC) and generalized policy iteration (GPI).

Receding-horizon CEC

CEC is a suboptimal control scheme that applies, at each stage, the control that would be optimal if the noise variables $\boldsymbol{w}_t$ were fixed at their expected values (zero in our case). The main attractive characteristic of CEC is that it reduces stochastic optimal control problem to a deterministic 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.

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

s.t.

$$\boldsymbol{e}_{t+1} = g(t, \boldsymbol{e}_t, \boldsymbol{u}_t, \boldsymbol{0})$$
$$\boldsymbol{u}_t \in \mathcal{U}$$
$$\tilde{\boldsymbol{p}}_t + \boldsymbol{r}_t \in \mathcal{F}$$

where, $\mathfrak{q}(\boldsymbol{e}_{\tau+T})$ is the terminal cost defined as

$$\mathfrak{q}(\boldsymbol{e}_{\tau+T}) = \tilde{\boldsymbol{p}}_{\tau+T}^T \boldsymbol{Q}\tilde{\boldsymbol{p}}_{\tau+T} + q\left(1-\cos\tilde{\theta}_{\tau+T}\right)^2$$

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

$$\min_{\boldsymbol{U}} c(\boldsymbol{U}, \boldsymbol{E})$$

s.t.

$$\boldsymbol{U}_{lb} \le \boldsymbol{U} \le \boldsymbol{U}_{ub}$$

$$\boldsymbol{h}_{lb} \le \boldsymbol{h}(\boldsymbol{U}, \boldsymbol{E}) \le \boldsymbol{h}_{ub}$$

where, $\boldsymbol{U} \coloneqq \left[\boldsymbol{u}_\tau^T, \dots, \boldsymbol{u}_{\tau+T-1}^T\right]^T$ refers to the sequence of control inputs and $\boldsymbol{E} \coloneqq \left[\boldsymbol{e}_\tau^T, \dots, \boldsymbol{e}_{\tau+T}^T\right]^T$ refers to the sequence of error states.

Now, the NLP program is solved by a popular NLP solver known as *CasADi*. Once a control sequence $\boldsymbol{u}_\tau, \dots, \boldsymbol{u}_{\tau+T-1}$ is obtained, CEC applies the first control $\boldsymbol{u}_\tau$ to the system to obtain the new error state $\boldsymbol{e}_{\tau+1}$ at time $\tau + 1$, and then repeats the optimization process for the given equation online in order to determine the control input $\boldsymbol{u}_{\tau+1}$. This online re-planning is necessary because the CEC formulation does not take the effect of motion noise into account.

NLP formulation using CasADi

CasADi is an open-source tool for non-linear optimization and algorithmic differentiation. It facilitates efficient implementation of different methods for numerical optimal control, both in an offline context and or non-linear model predictive control (NMPC). NLPs can be solved using CasADi using a specific block structure or general sparsity exploiting sequential quadratic programming (SQP) or interfaces to IPOPT.

The NLP for the tracking problem is formulated using CasADi's $Opti()$ stack in python. The elements of the NLP formulation are described as follows.

First, a function $cec\_controller()$ is defined where we initiate an $Opti()$ object as the optimizer. Value for τ is taken as zero. The time horizon T is taken as a constant for the problem which incorporates the long-term cost while solving the problem iteratively and it is initialized as one. The time step Δ is taken as 0.5.

Optimization variables:

The control inputs $\boldsymbol{u} \coloneqq \left[\boldsymbol{u}_0^T, \dots, \boldsymbol{u}_{T-1}^T\right]^T$ and error states $\boldsymbol{e} \coloneqq \left[\boldsymbol{e}_0^T, \dots, \boldsymbol{e}_T^T\right]^T$ are defined as the variables that are to be optimized using $Opti.variable()$. The first error state is initialized as $current\_state - current\_reference$.

Parameters:

The different parameters in the problem are defined using $Opti.parameter()$. These are initialized using $Opti.set\_value()$ as follows.

- $\boldsymbol{Q} = \mathbb{I}^{2\times 2}$
- $\boldsymbol{R} = \mathbb{I}^{2\times 2}$
- $q = 1$
- $\gamma = 1$

The reference trajectory $ref \in \mathbb{R}^3$ is obtained at each time step using the function $lissajous()$ that also takes in $current\_iteration$ as an input.

Constraints:

The different constraints and boundary conditions for the NLP are set using $Opti.subject\_to()$. The constraints for controls $\boldsymbol{u}_t \in \mathcal{U}$ are defined as

- $Linear\ velocity, v \in [0,1]$
- $Angular\ velocity, \omega \in [-1,1]$

The constraints for robot space coordinates $(x, y) = \tilde{\boldsymbol{p}}_{\boldsymbol{t}} + \boldsymbol{r}_{\boldsymbol{t}} \in \mathcal{F}$ are defined as

- $-3 \leq x \leq 3$
- $-3 \leq y \leq 3$
- $(x+2)^2 + (y+2)^2 > 0.5^2$
- $(x-1)^2 + (y-2)^2 > 0.5^2$

The constraint for the error $\boldsymbol{e}_{\boldsymbol{t+1}} = g(t, \boldsymbol{e}_{\boldsymbol{t}}, \boldsymbol{u}_{\boldsymbol{t}}, \boldsymbol{0})$ is also set.

Objective function:

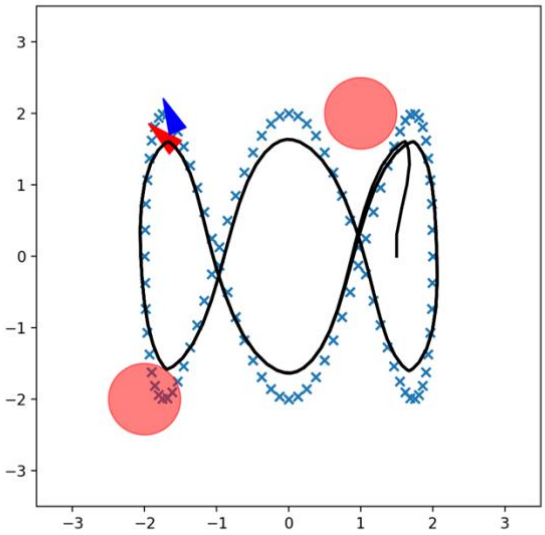
The objective function for V^* is defined and it is set to minimized by the optimizer using $Opti.minimize()$.

After the NLP formulation is complete, the optimization is performed by setting $Opti.solver()$ to the popular optimizer IPOPT and finally using $Opti.solve()$ to solve the NLP problem and compute all the required variables. The predicted controls for just the next step $\boldsymbol{u}_{\boldsymbol{\tau+1}}$ are then returned for the executing the robot motion using that input at the next instant.

IV. Results

We implement reduced-horizon CEC to obtain the results for the trajectory tracking problem for the robot for zero motion noise as the CEC approach is meant for deterministic case. The results can be expressed qualitatively in the form of trajectory visualizations as Lissajous figures and quantitatively in terms of the execution time and final error with respect to reference trajectory. We experiment with the NLP parameters for γ, q, Q, R to tune them for the optimal performance of robot for the best possible trajectory tracking.

- $\gamma = 1, q = 1, Q = \mathbb{I}^{2\times2}, R = \mathbb{I}^{2\times2}$

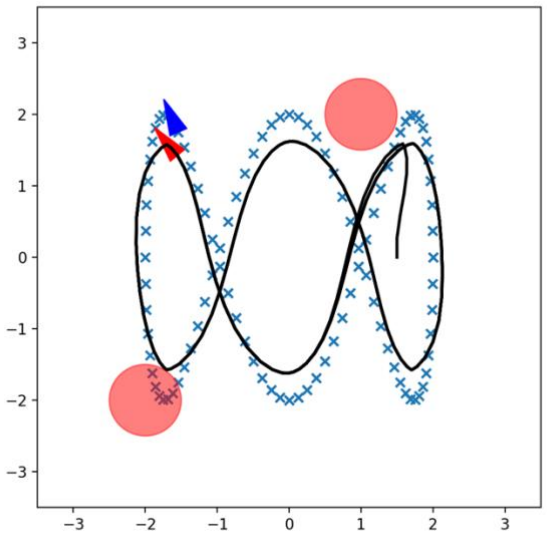


Total time: 16.474835872650146

Average iteration time: 68.59946846961975 *ms*

Final error: 132.23667742811864

- $\gamma = 0.9, q = 1, Q = \mathbb{I}^{2\times2}, R = \mathbb{I}^{2\times2}$

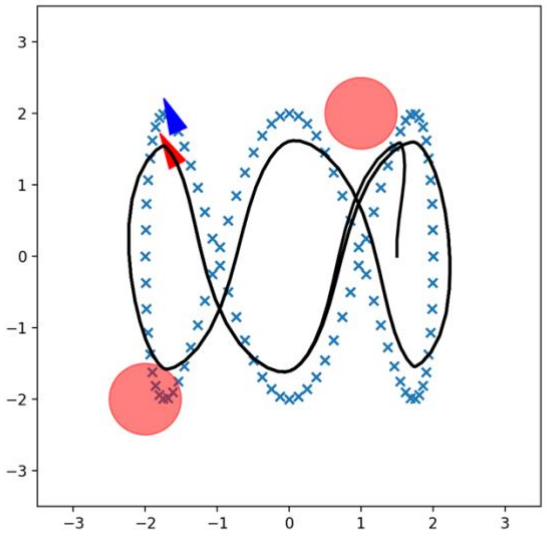


Total time: 16.605098009109497

Average iteration time: 69.14215882619222 *ms*

Final error: 111.2712293152787

- $\gamma = 0.8, q = 1, Q = \mathbb{I}^{2\times2}, R = \mathbb{I}^{2\times2}$

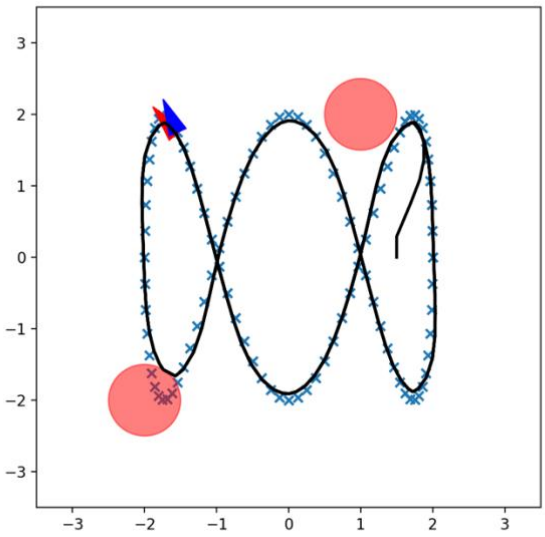


Total time: 16.905534029006958

Average iteration time: 70.39558589458466 *ms*

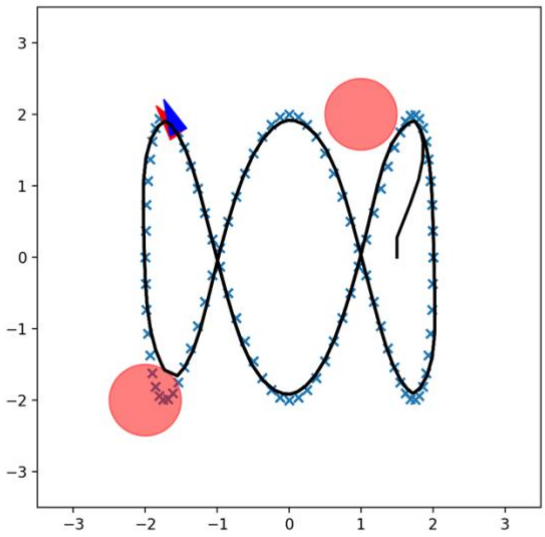
Final error: 118.80375943027127

- $\gamma = 0.9, q = 1, Q = 10 \times \mathbb{I}^{2\times2}, R = \mathbb{I}^{2\times2}$



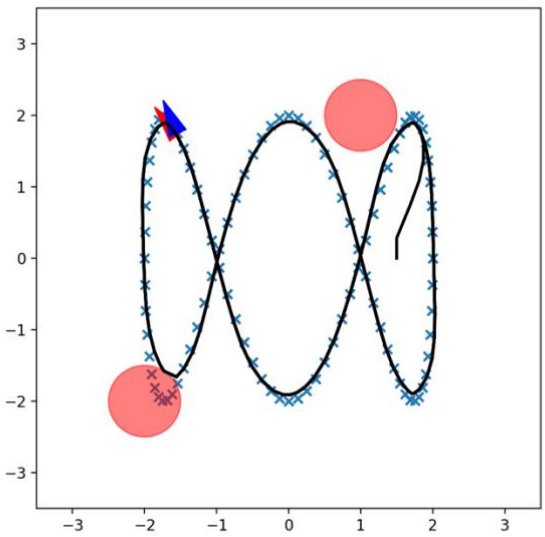
Total time: 17.29372525215149
Average iteration time: 72.01215823491415 *ms*
Final error: 112.33983775203038

- $\gamma = 0.9, q = 20, Q = 10 \times \mathbb{I}^{2\times2}, R = \mathbb{I}^{2\times2}$



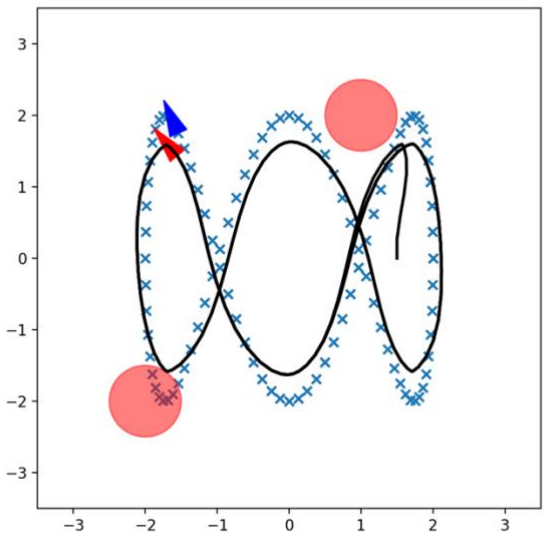
Total time: 17.164419174194336
Average iteration time: 71.47578001022339 *ms*
Final error: 108.96673313254196

- $\gamma = 0.9, q = 10, Q = 10 \times \mathbb{I}^{2\times2}, R = \mathbb{I}^{2\times2}$



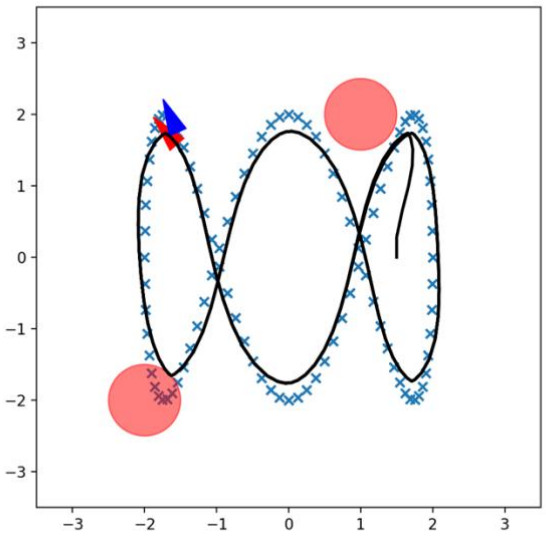
Total time: 17.0155189037323
Average iteration time: 70.85386912027995 *ms*
Final error: 110.57480288135436

- $\gamma = 0.9, q = 20, Q = 10 \times \mathbb{I}^{2\times2}, R = 10 \times \mathbb{I}^{2\times2}$



Total time: 16.539000034332275
Average iteration time: 68.86983017126718 *ms*
Final error: 110.24108382800199

- $\gamma = 0.9, q = 20, Q = 10 \times \mathbb{I}^{2\times2}, R = 5 \times \mathbb{I}^{2\times2}$

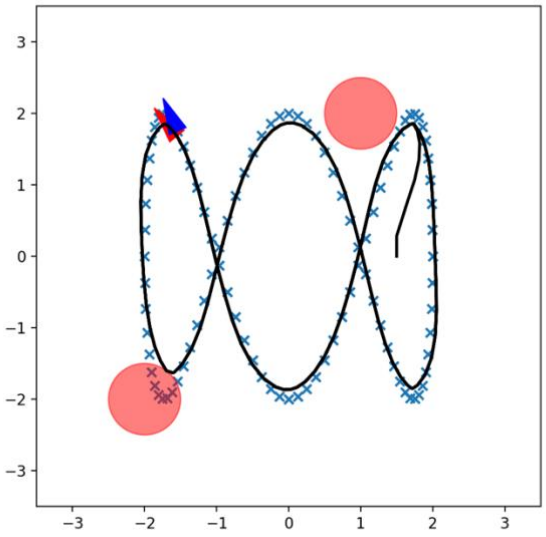


Total time: 16.78248405456543

Average iteration time: 69.88271772861481 *ms*

Final error: 105.55502745862074

- $\gamma = 0.9, q = 20, Q = 10 \times \mathbb{I}^{2\times2}, R = 2 \times \mathbb{I}^{2\times2}$

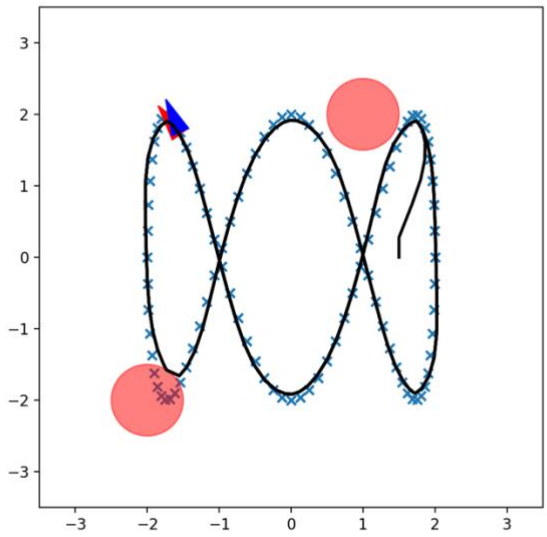


Total time: 16.889557123184204

Average iteration time: 70.33006250858307 *ms*

Final error: 108.16189417506178

- $\gamma = 0.9, q = 40, Q = 20 \times \mathbb{I}^{2\times2}, R = 2 \times \mathbb{I}^{2\times2}$



Total time: 17.31428360939026

Average iteration time: 72.09790448347728 *ms*

Final error: 108.89822040753835

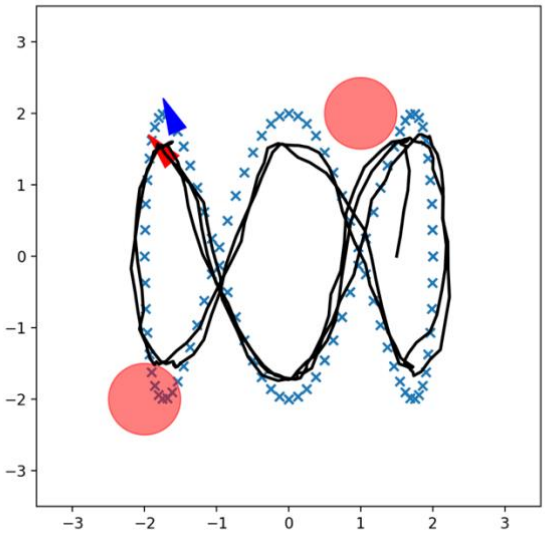
We can infer a lot about how the parameters γ, q, Q, R affect the trajectory tracking performance based on the plots and the error values.

- We start with $\gamma = 1$ and notice that performance is quite reasonable in terms of tracking, but we have a high trajectory error. Thus, reduce it slightly to 0.9 and notice that the error reduces. Lowering its value any further drastically affects the performance. This is because the performance will be better if we consider the future reference points more while tracking instead of being very short-sighted. So, we fix $\gamma = 0.9$.
- We increase the Q factor responsible for the cost of deviating from reference position trajectory to 10 and notice that the tracking is very accurate, along with a lower error. This is because increasing the Q value will make sure that the path traced by robot is much closer to the reference trajectory as it would try to minimize this error owing to a higher weightage.
- We then increase the q factor responsible for the cost of deviating from the reference orientation trajectory to 10, same as Q value. We notice that the trajectory is accurate and the error is also reduced further. This is because we are now prioritizing the orientation along with the position. Increasing the q value to 20 further reduces the error and we notice that the robot takes sharper turns as it tries to stick to the reference orientation with much more emphasis.
- We then experiment with the R factor responsible for the cost for using excessive control effort. We set

it to 10 and we notice that the error increases and the trajectory gets worse as well. Setting it to 5 makes the trajectory slightly deviated from the corners and the robot tries to deal the obstacles in a much safer manner by turning towards the future reference in a much smoother manner. Thus, we get the least error in this case. This happens because the robot tries to emphasize the controls due to higher error weightage and a relatively lesser emphasis to the reference trajectory. Reducing it further to 2 makes the trajectory much more accurate to the reference but increases the error slightly. In this case, the robot tries to follow the reference as much as it can and takes turns abruptly at the very last moment, inducing higher error.

We also try to implement the CEC approach for the tracking problem with the motion noise. We try different parameters and pick one possible combination to show here.

- $\gamma = 0.9, q = 10, Q = 10 \times \mathbb{I}^{2\times2}, R = 10 \times \mathbb{I}^{2\times2}$



$Total\ time$: 16.52368712425232
$Average\ iteration\ time$: 68.80231897036235 ms
$Final\ error$: 115.90912588310562

We notice that a lot of combinations for the parameters lead to infeasible solutions. We observe that the trajectory tracking is not that accurate, and the robot motion is quite erratic which is obvious due to the presence of noise during motion. So, the CEC method does not do well if the motion noise is present and implementing the GPI method would be much appropriate for handling such a case.