Infinite-Horizon Stochastic Optimal Control using CEC and GPI Approaches

Sanchit Gupta
Department of Electrical and
Computer Engineering
University of California San Diego
sag006@ucsd.edu

***Abstract*—This report discusses about the approaches to design a control policy for the ground differential-drive robot in order to track a desired reference trajectory while avoiding collisions with obstacles in the environment. The trajectory tracking problem is formulated as a discounted infinite-horizon stochastic optimal control problem. It is solved using the approaches of receding-horizon certainty equivalent control (CEC) and generalized policy iteration (GPI). The results of trajectory tracking performed by robot using both the approaches in the given environment are presented in this report.**

Keywords—Differential-drive robot, Control policy, Stochastic optimal control, Infinite-horizon, CEC, GPI

I. INTRODUCTION

Motion planning is an integral function for the autonomous navigation of the robot. It is essential to enable the robot to navigate autonomously from a given location to the target location in the shortest path possible in any environment, while avoiding the obstacles as present in the path. In real-life scenarios, the robot will have to navigate by creating its own control policy dynamically using the map of the environment and taking feedback from the sensors. The possibilities of incomplete map of the environment, disturbance in sensor measurements due to noise from various sources and other external factors affecting the movement of the robot give rise to stochasticity in the model.

The approach to address the stochasticity in the model requires to reformulate a deterministic optimal control problem into a stochastic optimal control problem. The motion model of the robot will now consist of a probability distribution to determine the state estimations. Due to the noise embedded in the model, the control policy may not always converge to the target position in finite number of steps and hence, the problem is converted to infinite horizon to allow the policy to converge. Thus, in real-life scenarios, the motion planning for navigation of a robot becomes a problem of infinite-horizon stochastic optimal control.

Dynamic Programming algorithms help in computing the policies for the infinite-horizon stochastic optimal control problem. It exploits the Bellman equation to convert the problem to discounted stochastic optimal control, ensuring convergence to optimal policy as horizon tends to infinity. Trajectory tracking is one such application of infinite-horizon stochastic optimal control problems which find various applications in robotics in the fields of warehouse robotics, robotic arm manipulators, autonomous driving and others.

This project requires us to design a control policy for a ground differential-drive robot for safely tracking a reference trajectory while avoiding obstacles. The robot follows a discrete-time kinematic motion model with motion noise modelled as a Gaussian distribution. The given environment is a map of $[-3,3]^2$ with two circular obstacles C_1 centered at $[-2,-2]$ and C_2 centered at $[1,2]$ and with radius 0.5. The error state is defined to measure the deviation of position and orientation from the reference trajectory. The trajectory tracking problem with time horizon and tracking error state is formulated as discounted infinite-horizon stochastic optimal control problem. The problem is to be solved using two approaches of receding-horizon certainty equivalent control (CEC) and generalized policy iteration (GPI). The technical approach and results for both of these approaches are discussed further in this report.

II. PROBLEM FORMULATION

This section will discuss about the formulation of infinite-horizon stochastic optimal control problem for trajectory tracking of a robot and its implementation in terms of mathematical formulas. The following details are provided in the project:

- The environment is given as a map of $[-3,3]^2$ with two circular obstacles C_1 centered at $[-2,-2]$ and C_2 centered at $[1,2]$. The radius of both the obstacles is 0.5. $F = [-3,3]^2 \setminus C_1 \cup C_2$ denotes the free space of the environment.
- The desired reference trajectory has position $r_t \in R^2$ and orientation trajectory $\alpha_t \in [-\pi,\pi]$. The reference trajectory is designed as a lissajous curve with time period T = 100.
- The robot's state is defined as $x_t = (p_t, \theta_t)$ at discrete-time $t \in N$ consisting of position $p_t \in R^2$ and orientation $\theta_t \in [-\pi,\pi]$. The robot is controlled using velocity input $u_t = (v_t, \omega_t)$ consisting of linear velocity $v_t \in R$ and angular velocity $\omega_t \in R$.

- The control input u_t is limited to an allowable set of linear and angular velocities $U = [0,1]\ x\ [-1,1]$.
- The discrete-time kinematic motion model of the differential-drive robot with time interval $\Delta > 0$ is given in equation 1.

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

- In equation 1, $w_t \in R^3$ models the motion noise with Gaussian distribution $N(0, diag(\sigma)^2)$ with standard deviation $\sigma = [0.04, 0.04, 0.004] \in R^3$. The noise is assumed to be independent across time and robot state x_t.
- The error state is defined as $e_t = (\tilde{p}_t, \tilde{\theta}_t)$ 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. Equation 2 shows the equation of motion for error dynamics.

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

An image of the environment with reference trajectory and obstacles is shown in Figure 1. The blue arrow in the figure indicates the robot following the desired trajectory (*indicated using cyan crosses*) and the red arrow indicates the robot which would follow the trajectory as per the controls given. The circular objects in pink resemble the two obstacles.

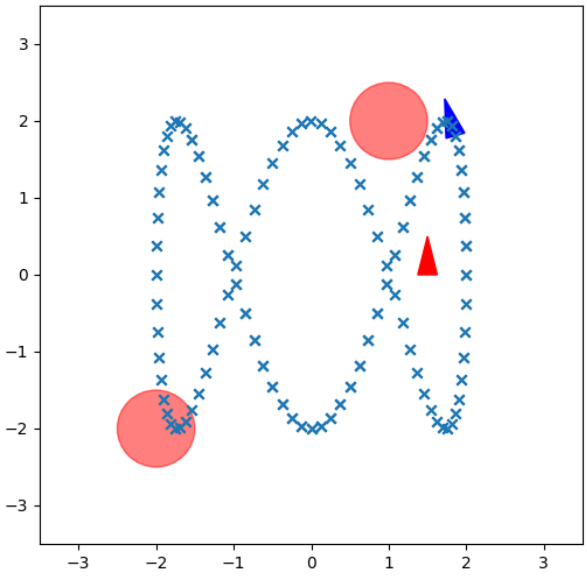

Figure 1: An image of the environment with the reference trajectory and obstacles

The trajectory tracking with initial time τ and initial error $\boldsymbol{e}$ is now formulated as a discounted infinite-horizon stochastic optimal control problem, as shown in equation 3.

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

The problem formulated in equation 3 is subjected to many constraints which are given in equation 4.

$$\begin{aligned} &e_{t+1} = g(t, e_t, u_t, w_t), \qquad w_t \sim N(0, diag(\sigma)^2) \\ &u_t = \pi(t, e_t) \in U \\ &\tilde{p}_t + r_t \in F \end{aligned} \quad (4)$$

The symbols and constants used in equation 3 are defined as follows:

- $Q \in R^{2x2}$ is a symmetric positive-definite matrix defining the stage cost for deviating from the reference position trajectory r_t
- $q > 0$ is a scalar defining the stage cost for deviating from the reference orientation trajectory α_t
- $R \in R^{2x2}$ is a symmetric positive-definite matrix defining the stage cost for using excessive control effort

The objective of the project is to solve equation 3 subject to constraints given in equation 4 and obtain an optimal control policy to enable the robot follow the desired trajectory. The problem defined in equation 3 will be solved using two different approaches of receding-horizon certainty equivalent control (CEC) and generalized policy iteration (GPI). Using this formulation, the implementation of CEC and GPI approaches to solve the problem defined is discussed in next section of the report.

III. Technical Approach

The technical approach to solve the trajectory tracking problem formulated as discounted infinite-horizon stochastic optimal control problem in Section II is discussed in detail in this section. The approaches of CEC and GPI to solve this problem is elaborated in different sub-sections below.

A. Part 1: Receding-horizon certainty equivalent control (CEC)

CEC is a suboptimal control scheme that applies the control which is found to be optimal at each stage. This would be applicable if the noise w_t is fixed at their expected value. The expected value of noise is zero in our implementation, as defined in equation 4. The characteristic of CEC is that it reduces the stochastic optimal control problem into deterministic optimal control problem by discarding the noise variable.

As CEC converts the problem to deterministic optimal control problem, the problem now reduces to finite-horizon from infinite-horizon. CEC approximates by repeatedly

solving the discounted finite-horizon deterministic optimal control problem at each time step and thus, equation 3 converts to equation 5 in the implementation of CEC.

$$V^*(\tau, e) = \min_{e_\tau,\dots,e_{\tau+T-1},\, u_\tau,\dots,u_{\tau+T-1}} q(e_{\tau+T}) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau} \left(\tilde{p}_t^T Q \tilde{p}_t + q\left(1 - \cos\left(\tilde{\theta}_t\right)\right)^2 + u_t^T R u_t \right) \quad (5)$$

Equation 5 is subjected to the constraints as given in equation 6.

$$\begin{aligned} & e_{t+1} = g(t, e_t, u_t, 0), \qquad t = \tau, \dots., \tau + T - 1 \\ & u_t \in U \\ & \tilde{p}_t + r_t \in F \end{aligned} \quad (6)$$

$q(e_{\tau+T})$ is the terminal cost defined as given in equation in 7 for the terminal time T.

$$q(e_{\tau+T}) = \tilde{p}_{\tau+T}^T Q \tilde{p}_{\tau+T} + q\left(1 - \cos\left(\tilde{\theta}_{\tau+T}\right)\right)^2 \quad (7)$$

After formulating the CEC problem in equation 5 and constraints in equation 6, the receding-horizon CEC problem now converts into a non-linear problem (NLP) of the form as shown in equation 8.

$$\begin{aligned} & \min_U c(U, E) \\ & such\ that\ U_{lb} \le U \le U_{ub} \\ & h_{lb} \le h(U, E) \le h_{ub} \end{aligned} \quad (8)$$

Where, $U = [u_\tau^T, \dots, u_{\tau+T-1}^T]^T \; and \; E = [e_\tau^T, \dots, e_{\tau+T}^T]^T$ and the bounds for $U = [0,1]\; x\; [-1,1]$. Further, h is any non-linear function here which captures the constraints in equation 6.

Once the control sequence $U = [u_\tau^T, \dots, u_{\tau+T-1}^T]^T$ is obtained, CEC applies the first control u_τ to the system, calculates the new error state $e_{\tau+1}$ at time $\tau + 1$ and repeats this optimization process to determine the new control input $u_{\tau+1}$.

A-1: Formulation to solve NLP problem using CasADi

CasADi is an open-source tool for solving the non-linear optimization problems. It provides a framework to present the problem in symbolic form. Since the CEC problem is now converted into a non-linear problem (NLP), the NLP solver of CasADi helps in minimizing the equation 5 by solving it iteratively. The minimum value function is the optimal value function V^* and the controls at which the equation is minimized is the optimal control policy for the model at a give time-step.

The Opti stack of CasADi is used to formulate the whole problem in the code. The details of the code formulation are described below:

- The optimizer is initialized by calling casadi.Opti() function.
- The time horizon T is a constant parameter. It can be varied for any value greater than 0 for solving the problem iteratively. $T = 1$ was used initially to check if the CEC controller was functioning appropriately. The value of T was then increased to different values to understand the efficient functioning of the controller.
- $U = [u_\tau^T, \dots, u_{\tau+T-1}^T]^T \; and \; E = [e_\tau^T, \dots, e_{\tau+T}^T]^T$ are the variables to be optimized. These are initialized and set as variables using opti.variable().
- All the other parameters in equation 5 which are not to be optimized are initialized using opti.parameter() and their value is set using opti.set_value(). The initial values of all such parameters are given below:
  - Q is initialized as an Identity matrix of dimension $2x2$.
  - R is initialized as an Identity matrix of dimension $2x2$.
  - q is initialized as a scalar with value = 1.
  - The discount factor γ is initialized as 0.5.
- The reference position states $r_t \in R^2$ and orientation states $\alpha_t \in [-\pi, \pi]$ are also initialized as parameter using opti.parameter().
  - For all the timesteps within the time horizon T in the steps of $time\_step = 0.5$, the values of reference states is set by calling the function 'lissajous()' defined in main.py file.
- Once the variables and parameters are set, it becomes important to set the constraints for the NLP solver. Opti.subject_to() is used to set all the boundary conditions and constraints. Following constraints are set on variables for the solver:
  - $linear\ velocity \in [0,1]$
  - $angular\ velocity \in [-1,1]$
  - For the new state as per a given control $(x, y) = \tilde{p}_t + r_t$ and $\theta = \tilde{\theta}_t + \alpha_t$,
    - $-3 \le x \le 3$
    - $-3 \le y \le 3$
    - $-\pi \le \theta < \pi$
  - In order to avoid the obstacles, the following constraints are set for (x, y)
    - $(x+2)^2 + (y+2)^2 > (0.5)^2$ to avoid C_1
    - $(x-1)^2 + (y-2)^2 > (0.5)^2$ to avoid C_2
- As $E = [e_\tau^T, \dots, e_{\tau+T}^T]^T$ is a variable to be optimized, equation 2 given above will also be a constraint equation to set using opti.subject_to().
- After setting all the boundary conditions and constraints, the equation 5 to be minimized is set in the code. It is given to opti.minimize() function for

minimizing it and for solving for the variables defined.

Once the problem is formulated in code for CasADi, the NLP solver is called to minimize the set equation and calculate the variables. IPOPT is one of the popular NLP solver interfaced with CasADi which is called using opti.solver() function to solve the NLP problem.

CasADi NLP solver solves the problem iteratively and repeats the optimization process online to determine the new control input $u_{\tau+1}$. This online re-planning is important and necessary as CEC controller does not take the motion noise into account during the formulation. The results of trajectory tracking using CEC controller is shown in section IV.

B. Part 2: Generalized Policy Iteration (GPI)

GPI requires implementation of either value iteration process or policy iteration process to determine the optimal value function and optimal policy at a given time-step. As the state space and control space are continuous, it is important to discretize them into a number of grid points appropriately. The state space can be discretized into $(n_t, n_x, n_y, n_\theta)$ number of grid points. At the starting of the formulation, the state space is discretized coarsely as follows:

- $n_t = 100$ as the provided reference trajectory is periodic with period of 100
- $n_x = 13$ to discretize $x \in [-3,3]$ in steps of 0.5
- $n_y = 13$ to discretize y $\in [-3,3]$ in steps of 0.5
- $n_\theta = 13$ to discretize $\theta \in [-\pi, \pi)$ in steps of 30 degrees

In addition, the control space can be discretized into (n_v, n_w) number of grid points. The control space is also discretized coarsely at the start of the formulation as follows:

- $n_v = 11$ to discretize $v \in [0,1]$
- $n_v = 21$ to discretize $\omega \in [-1, 1]$

B-1: Formulation for GPI

The equation 3 as given above with the constraint conditions as given in equation 4 is to be minimized to obtain the optimal value function and optimal control policy at a given time-step. It becomes necessary to convert equation 5 into Bellman equation for discounted problem and solve using the methods of Value Iteration or Policy Iteration. Since the problem is of infinite horizon, the terminal cost term will not be included in the problem as the problem may not always converge to the terminal state. Using Bellman equation, equation 3 now converts to discounted problem with optimal value function as shown in equation 9.

$$V^*(\tau, e) = \min_{u \in U} E\left[\gamma\left(\tilde{p}_t^T Q \tilde{p}_t + q\left(1 - \cos\left(\tilde{\theta}_t\right)\right)^2 + u_t^T R u_t\right)\middle|\ e_\tau = e\right] \quad (9)$$

Further, the problem is initially solved deterministically and hence, the transition probabilities in the discretized space are not considered initially. Thus, it is considered that the robot will transit to the next state as calculated using the error state and given control with probability = 1.

The value iteration method is used to solve equation 9. As per the discretized control space, it would apply each control to the robot, calculate the error state for each control and thus, calculates the value function for each control. It then determines the control for which the value function is minimum. This process continues iteratively till the difference between two subsequent value function values falls below a threshold. The condition for convergence is given in equation 10.

$$|V_{k+1}(x) - V_k(x)| < \varepsilon \quad (10)$$

ε is considered as 0.001 initially to keep a stricter condition for convergence. After convergence, a list of controls is obtained from which the first control is taken passed to the main function.

The initial values of all the parameters to be tuned are given below:

- Q is initialized as an Identity matrix of dimension $2x2$.
- R is initialized as an Identity matrix of dimension $2x2$.
- q is initialized as a scalar with value = 1.
- The discount factor γ is initialized as 0.5.

Once the error is calculated for a given control, the new state for the robot is determined using $\tilde{p}_t + r_t$. It is possible that the new state calculated would not be according to the discretized state space. Since the transition probability matrix is not used for the deterministic case, the new state space is rounded off to the nearest state in the discretized state space and the error is then modified accordingly.

Conditions for obstacle avoidance: As the state space and controls are discretized within the boundary conditions, they are bound to follow $(x, y) \in [-3,3]^2$; $\theta \in [-\pi, \pi)$ and $U \in [0,1]x\ [-1,1]$ respectively. Further, the conditions for obstacle avoidance for (x, y) are set as below:

- $(x+2)^2 + (y+2)^2 > (0.5)^2$ to avoid C_1
- $(x-1)^2 + (y-2)^2 > (0.5)^2$ to avoid C_2
- For any state falling within the obstacle, the value function for it is set as infinity and hence, such a control is avoided.

The discretization was made dense for further iterations, after setting up the code initially. The results for different discretization of state and control spaces are shown in section IV.

IV. RESULTS

This section discusses about the results as obtained for trajectory tracking using CEC and GPI controllers for different sets of parameters $Q, R, q \; and \; \gamma$.

A. Results for Part 1 – CEC Controller

The initial tests were conducted to understand the variation in trajectory as the parameters $Q, R, q \; and \; \gamma$ were varied. Each of these parameters were increased while keeping the others at a low value. This helped in understanding the variation brought in by each parameter.

The controller was first run at the initial set of parameters to understand the behavior of controller initially. The trajectory obtained at the initial set of parameters is shown in Figure 2. All other details about this test are shown in Table 1.

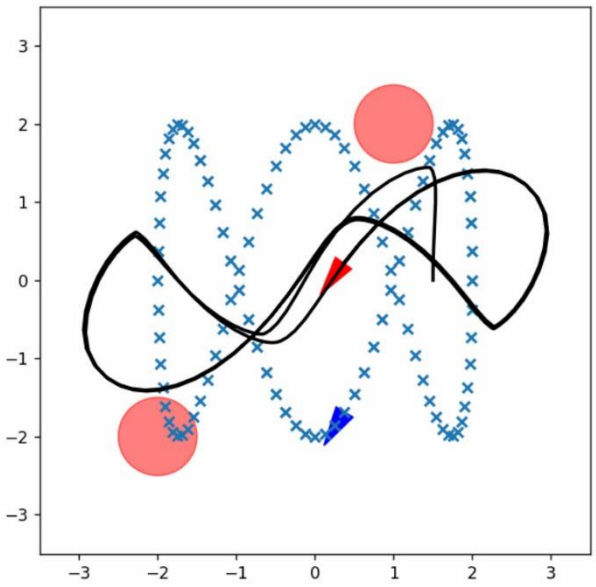

Figure 2: Trajectory tracking using CEC controller for initial set of parameters

After this, the value of γ was increased to 0.9, while keeping other three parameters to their original value. The trajectory tracking obtained for this test is shown in Figure 3. Increasing the value of γ improved the performance of the controller manifolds, however it induced abrupt turns in the trajectory of robot. This test helped in deciding that γ = 0.9 is an optimal value to consider for further tests. It was further experimented by increasing γ to 1 and decreasing it any other value between 0.5 and 0.9. In both the cases, the final error was found to be increasing.

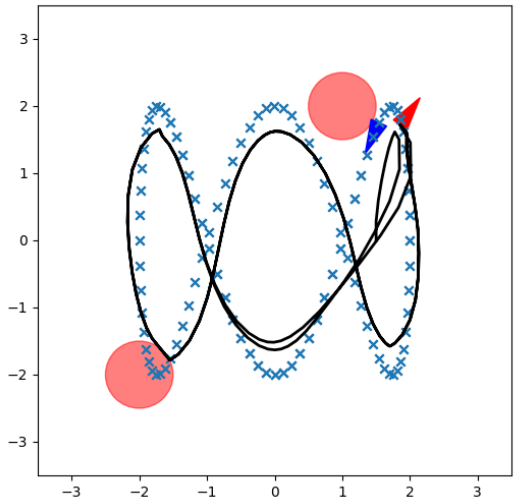

Figure 3: Trajectory tracking using CEC controller for γ = 0.9 and other parameters set at initial values

After finalizing γ, the value of Q was increased to 5 times the Identity matrix while the other parameters were kept as constant. This test showed that increasing the value of Q made the controller decrease the position error. However, since the other parameters were not optimized, the robot took abrupt turns in the navigation as well as the controls applied were not optimal to follow the desired trajectory. The trajectory obtained in this case is shown in Figure 4.

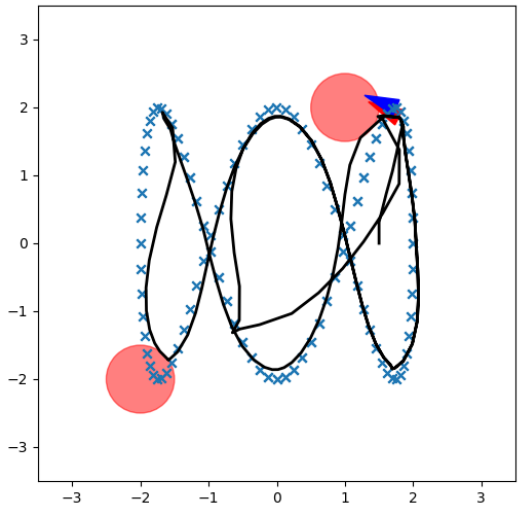

Figure 4: Trajectory tracking using CEC controller for Q = 5* Identity matrix

Further to understanding the behavior of Q matrix, the value of R matrix was increased. It was observed that the trajectory shrinks on increasing the value of R matrix to 5*Identity matrix. It can be inferred that the controller did not apply effort in minimizing the position or orientation error in this case while it applied more efforts to choose the best possible control input. The trajectory obtained in this case is shown in Figure 5.

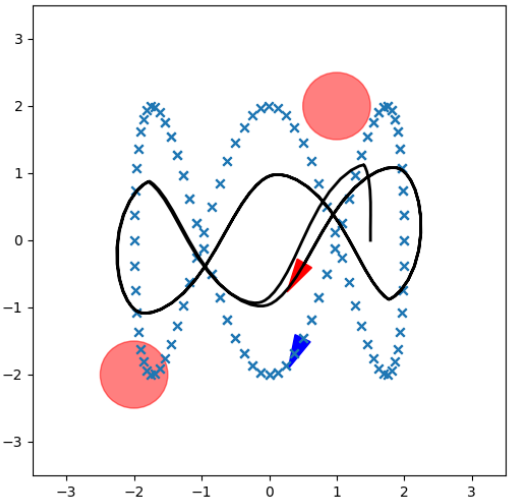

Figure 5: Trajectory tracking using CEC controller for R = 5* Identity matrix

Lastly, the value of parameter q was increased while keeping the other parameters as constant. It can be observed from this test that the controller tried to minimize the errors in orientation. However, as the parameters to minimize position error and for control effort were not tuned, the robot took many abrupt turns and the less optimal controls were applied which were not optimal to follow the desired trajectory. The trajectory obtained in this case is shown in Figure 6.

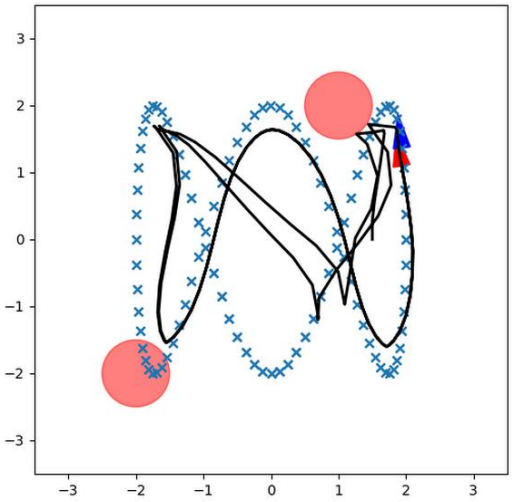

Figure 6: Trajectory tracking using CEC controller for q = 5

Basis the results obtained in these tests, the parameters were optimized in order to minimize the error to the extent possible and to get the optimal trajectory for following the desired trajectory. After tuning the parameters, it was found that the following set of parameters gave an optimal trajectory with low error.

- Q = 15*Identity matrix
- R = 15*Identity matrix
- q = 40
- Gamma = 0.9

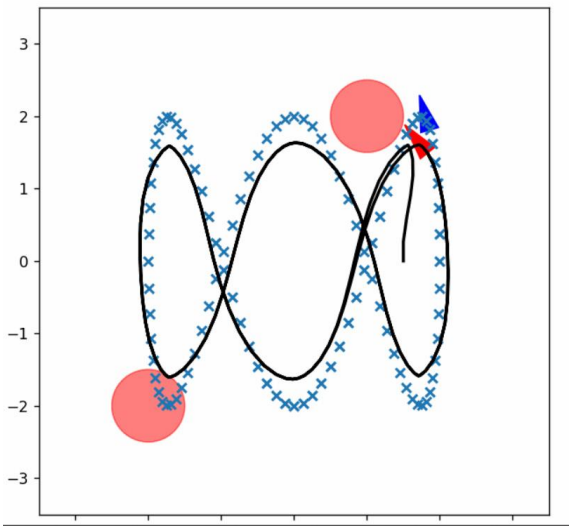

Figure 7: Trajectory tracking for one set of optimal parameters after tuning while ignoring the noise

Figure 7 shows the trajectory tracking for the optimal set of parameters described above. This result was obtained while ignoring the noise. In addition, the experiments were conducted to tune the parameters and determine the optimal set of parameters while considering the noise in the calculation of next state of the robot. It was quite difficult to obtain parameters in this case as many parameters gave infeasible solution. However, for the following set of parameters, an optimal trajectory was found with minimal error.

- Q = 10*Identity matrix
- R = 15*Identity matrix
- q = 5
- Gamma = 0.9

The trajectory obtained in this case is shown in Figure 8.

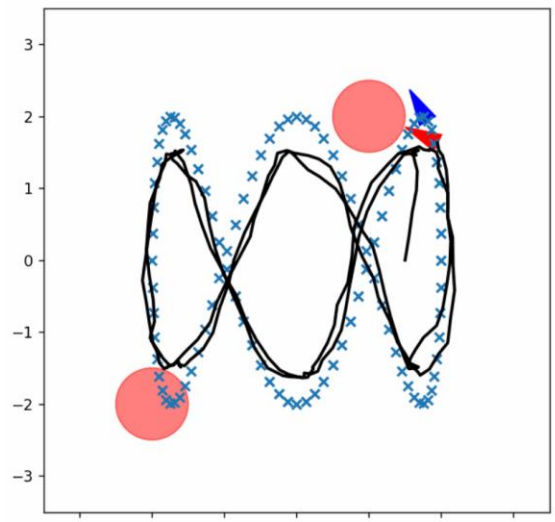

Figure 8: Trajectory tracking for one set of optimal parameters after tuning while considering the noise

The details of all the tests conducted for the CEC controller are summarized in Table 1.

Table 1: Details of tests conducted for CEC controller

Q	R	q	γ	Total time (in sec)	Avg. time per iteration (in ms)	Error (Units)	Figure
I	I	1	0.5	23.9	97.57	455.82	2
I	I	1	0.9	16.07	66.97	155.56	3
5*I	I	1	0.9	16.54	68.93	273.0	4
I	5*I	1	0.9	19.3	80.41	275.5	5
I	I	5	0.9	15.5	64.66	328.6	6
15*I	15*I	40	0.9	12.27	47.44	109.3	7
10*I	15*I	10	0.9	13.3	49.5	120.2	8

B. Results for Part 2 – GPI Controller

The initial test for GPI controller was conducted at the initially discretized state and control spaces and for initial set of parameters, as mentioned in the technical approach. The trajectory obtained in this case is shown in Figure 9.

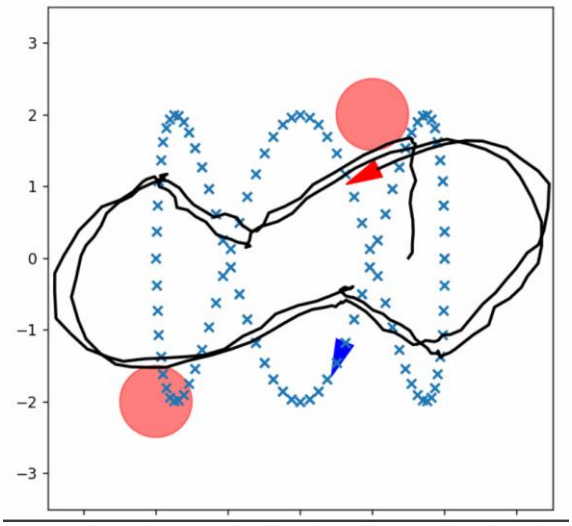

Figure 9: Trajectory tracking obtained using GPI controller for initial set of parameters and discretization

Further to this test, the discretization was doubled for state and control spaces. They were made dense as described below:

- $n_t = 100$ as the provided reference trajectory is periodic with period of 100
- $n_x = 26$ to discretize $x \in [-3,3]$ in steps of 0.25
- $n_y = 26$ to discretize y $\in [-3,3]$ in steps of 0.25
- $n_\theta = 26$ to discretize $\theta \in [-\pi, \pi)$ in steps of 15 degrees
- $n_v = 22$ to discretize $v \in [0,1]$
- $n_v = 42$ to discretize $\omega \in [-1, 1]$

The trajectory obtained in this case is shown in Figure 10.

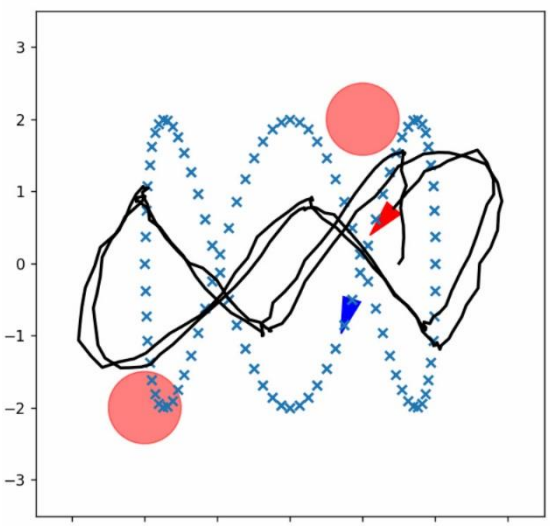

Figure 10: Trajectory tracking obtained using GPI controller for initial set of parameters but after doubling the discretization

On comparing Figure 9 and 10, it can be observed that the trajectory improved as the discretization was made dense. Due to time constraints, I could tune the parameters in this case.

Resulting Discussion

- For CEC controller, the effect of each cost parameter was studied by varying one of the parameters while keeping the others constant.
    - Increasing the value of γ enhanced the performance of the controller but induced some abrupt turns. It was finalized at 0.9.
    - Increasing the value of Q matrix helped the controller in reducing position deviation but the robot took abrupt turns and took less optimal controls for navigation.
    - Increasing the value of R made the trajectory shrink but the controller did not apply efforts in reducing position or orientation deviation.
    - Increasing the value of q made the controller reduce the orientation error.
- After studying the effects of each parameter, they were tuned to get the best possible trajectory. The parameters were different while ignoring the noise and while considering the noise in calculation of next state of the robot.
- It can be inferred that CEC controller is good for deterministic model whereas it is not the best controller to use for stochastic models with noise embedded in it. When the noise is considered, it was found that many parameters gave infeasible solutions as the noise would have made the robot enter into the obstacles.
- Lastly, the GPI controller is more suitable for stochastic models however, it is dependent on the level of discretization of the state and control spaces. Higher the discretization, more appropriate would be the trajectory tracking in this case.
- If I had more time, I would have:
    - Tried to optimize the GPI controller with increasing the level of discretization for state and control spaces.
    - Tried to optimize cost parameters for GPI controller.
    - Further tried to optimize the cost parameters for CEC controller to get more optimal trajectory with a lower error.