Project 3 - ECE 276B

Varun Pawar
Electrical and Computer Engineering
UC San Diego
La Jolla, CA
vpawar@ucsd.edu

***Index Terms*—Stochastic Optimal Planning, Policy Iteration, CEC**

I. Introduction

In the previous two projects, we successfully implemented motion plans for robots in discrete-deterministic settings. But, most modern robotics systems are stochastic in nature because of the uncertainty in sensor measurements. Here we solve the problem of planning optimal control when the system evolves stochastically in 3-dimensional space.

More specifically, the project requires generating a safe trajectory for a differential drive robot. The robot must track a predetermined trajectory while avoiding collision with a few obstacles simultaneously.

To complete the above objective, two different approaches are given. First is receding-horizon (CEC)certainty equivalent control. CEC is a sub-optimal deterministic method and it is supposed to be solved using non-linear programming using Casadi framework. The other approach is policy iteration in a known model case.

To summarize the objective:

- Implement optimal control in system evolving stochastically
- Make sure the proposed controller follows a described trajectory and avoids obstacles at the same time.
- Compare receding horizon CEC and GPI(generalized policy iteration) method using the experiments.

II. Problem Formulation

Consider a differtianl drive robot whose state is described as $\mathbf{x}_t := (\mathbf{p}_t, \theta_t)$, where $\mathbf{x}_t$ is the position of the robot in euclidean $\mathbb{R}^2$ space and θ_T is the orientation such that, $\theta_t \in \{-\pi, \pi\}$. The robot is controlled by a velocity input $\mathbf{u}_t = (v_t, \omega_t)$, where v_t and ω_t are linear and angular velocity respectively. The discrete-time kinematic model of the differential-drive robot obtained from Euler discretization of the continuous-time kinematics with time interval $\Delta > 0$ is:

$$\mathbf{x}_{t+1} = \begin{pmatrix} \mathbf{p}_{t+1} \\ \theta_{t+1} \end{pmatrix} + \begin{pmatrix} \Delta cos(\theta_t) & 0 \\ \Delta sin(\theta_t) & 0 \\ 0 & \Delta \end{pmatrix} \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + \mathbf{w}_t, \;\; t = 1, 2, \dots \tag{1}$$

where $\mathbf{w}_t$ is a motion noise with Gaussian distribution $\mathcal{N}(0, diag(\sigma^2))$ with standard deviation $\sigma = [0.04, 0.04, 0.004]$. The motion noise is assumed to be independent across time and of the robot state $\mathbf{x}_t$. The kinematics model in 1 defines the probability density function $p_f(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ of $\mathbf{x}_{t+1}$ conditioned on $\mathbf{x}_t$ and $\mathbf{u}_t$ as the density of a Gaussian distribution with mean $\mathbf{x}_t + \mathbf{G}(\mathbf{x}_t)\mathbf{u}_t$ and covariance $diag(\sigma)$.

The objective of the problem is to implement a control policy in the stochastic system evolving with dynamics given in 1 while tracking the trajectory given as $(\mathbf{r}_t, \alpha_t)$ where $\mathbf{r}_t$ is the reference position and α_t is the reference angle.

Moreover, $\mathbf{r}_t \in \mathbb{R}^2$ and $\alpha_t \in [-\pi, \pi]$. The robot also has to avoid the obstacle in the given space. There are two circular obstacles viz., $\mathcal{C}_1$ and $\mathcal{C}_2$, centered at $(-2, -2)$ and $(1, 2)$ respectively with radius 0.5.

Hence the free space is given as, $\mathcal{F} \in [-3, -3] \times [-3, 3] \setminus \mathcal{C}_1 \cup \mathcal{C}_1$.

A. Error space representation

Apart from the model established earlier, the robot can also be described using error state. Error state has advantage over global state space since it could be generalized to any possible trajectory. It is denoted as $\mathbf{e}_t = \mathbf{x}_t - \mathbf{r}_t$, and the associated system evolves as,

$$\mathbf{e}_{t+1} = \mathbf{e}_t + \begin{pmatrix} \Delta cos(\theta_t) & 0 \\ \Delta sin(\theta_t) & 0 \\ 0 & \Delta \end{pmatrix} \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + \begin{pmatrix} \mathbf{r}_t - \mathbf{r}_{t+1} \\ \alpha_t - \alpha_{t+1} \end{pmatrix} + \mathbf{w}_t \tag{2}$$

B. Control objective

Control objective is formulated as a trajecotry tracking problem. Given an initial time τ and initial error $\mathbf{e}$, the objective is formulated as a discounted infinite-horizon stochastic optimal control problem:

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

such that, $\mathbf{e}_{t+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t)$, $\mathbf{u}_t = \pi(t, \mathbf{e}_t) \in \mathcal{U}$ and $\mathbf{p}_t + \mathbf{r}_t \in \mathcal{F}$.

C. Cost Design:

Cost is designed as Euclidean distance($\mathbf{L2}$ norm).

III. Technical Approach

To solve the infinite-horizon stochastic optimal control objective, two different methods are implemented in this project.

A. Receding-horizon certainty equivalent control (CEC)

CEC is a sub-optimal 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. CEC reduces the stochastic objective to a fully deterministic optimal control problem. Moreover, it reduces the infinite horizon problem to a finite-horizon problem. Here, at every time step, a finite horizon optimal control objective is solved.

B. Formulation

The receding horizon CEC objective is written as,

$$\mathbb{V}^*(\tau, \mathbf{e}) = \min_{\mathbf{u}_\tau, \mathbf{u}_{\tau+2}, \ldots \mathbf{u}_{\tau+T-1}} \mathbf{q}(\mathbf{e}_{T+\tau}) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau}(\mathbf{p}_t^\mathbf{T}\mathbf{Q}\mathbf{p}_t + q(1-cost(\theta_t))^2 + \mathbf{u}_t^\mathbf{T}\mathbf{R}\mathbf{u}_t)$$

$$s.t.,$$
$$\mathbf{e}_{T+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0})$$
$$\mathbf{u}_t \in \mathcal{U}$$
$$\mathbf{e}_t + \mathbf{r}_t \in \mathcal{F}$$

The above problem can be solved using non-linear programming which is beyond the scope of this course. Therefore further discussions are on how to solve the problem using Casadi framework.

C. Casadi

Casadi is a symbolic equation solver framework available in python programming language. The framework is capable of solving non-linear programming objectives which as given below,

$$\min_{\mathbf{x}} \mathbf{f}(\mathbf{x}, \gamma) \quad given,$$
$$\mathbf{g}_{min} \leq \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_{max}$$

D. Converting CEC objective into Casadi NLP objective

Minimizing objective for NLP objective is written as,

$$\mathbf{f}(\mathbf{x}, \gamma) = \mathbf{V}^\pi(\tau, \mathbf{e}) \tag{4}$$

where $\mathbf{x} = \{\mathbf{e}_{\tau+1}, \mathbf{e}_{\tau+2}, \ldots, \mathbf{e}_{\tau+T-1}, \pi\}$ and $\gamma = \{\mathbf{Q}, \mathbf{R}, \mathbf{q}\}$. Constraint for the above NLP objective is written as,

$$\mathbf{g}(\mathbf{x}) = \{\{\mathbf{e}_{T+1} - g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0}) = 0\}, \{\mathbf{u}_t \in \mathcal{U}\}, \{\mathbf{e}_t + \mathbf{r}_t \in \mathcal{F}\}\} \tag{5}$$

E. Generalized Policy Iteration(GPI)

The next algorithm implemented in this project is generalized policy iteration. In order to implement GPI, the continuous state and control space is to be descretized into $(n_t, n_x, n_y, n_{theta})$ and (n_v, n_{omega}) number of grid points respectively. The state and control space is discretized over the region $[-3, 3]^2$ and $\mathcal{U}$. The transition probability in discretized MDP is for each discrete state $\mathbf{e}$ and each discrete control $\mathbf{u}$, we can choose the next grid points $\mathbf{e}^{'}$ around the mean $g(t,\mathrm{e},\mathrm{u},0)$, evaluate the likelihood of $\mathcal{N}(0, diag(\sigma^2))$ at the chosen grid points, and normalize so that the outgoing transition probabilities sum to 1.

F. Formulation and Implementation

The discretized formulation is given below,

$$\mathbf{e} = \{[x, y, \theta] | x \in \{-3, -3+dx, \ldots, 3-dx, 3\}, y \in \{-3, -3+dy, \ldots, 3-dy, 3\}, \theta \in \{-\pi, -\pi+d\theta, \ldots, \pi-d\theta\pi\}\}$$
$$\mathbf{u} = \{[v, \omega] | v \in \{0, 0+dv, \ldots, 1-dv, 1\}, \omega \in \{-1, -1+d\omega, \ldots, 1-d\omega, 1\}\}$$

The transition probability of next state is mean at,

$$\mathbf{e}_{next} = \mathcal{N}(g(t, \mathbf{e}, \mathbf{u}, 0), diag(\sigma^2))$$

In this problem, the value function is known therefore, the GPI algorithm reduces to policy evaluation.
During code implementation, it is assumed that there exists T such that $\gamma^T < \epsilon$. Therefore, the infinite time horizon reduces to a finite horizon case for a system.
In code implementation, policy iteration was implemented for $T = 3$ time horizon due to CPU bottleneck.

Algorithm 1 GPI algorithm

Require: State space set $\mathbf{E}$, control space set $\mathcal{U}$. Time horizon is T.
Ensure:
1: **while** True **do**
1: $\pi(t, e_t) = \min_{\pi} V^\pi(t, \mathbf{e}_t, \mathbf{r}_t, T)$
1: $t = t + 1$
2: **end while**=0

IV. Results

A. Metric

a. Tracking error: Tracking error is the euclidean distance between reference position and ground truth,

$$\mathbf{e}_{total} = \sum ||\mathbf{r} - \mathbf{x}||_2$$

$$\mathbf{e}_\mu = \sum \frac{1}{N} ||\mathbf{r} - \mathbf{x}||_2$$

b. Simulation time: Time taken to compute optimal control for one time-step.

Time Horizon(T)	$\mathbf{e}_{total}$	$\mathbf{e}_{\mu}$	$\mathbf{t}_{sim,Total}$(s)	$\mathbf{t}_{sim,\mu}$(ms)
1	579.3802	5.7387	10.1594	41.7372
3	125.9582	0.4054	10.3444	42.4629
5	105.0396	0.4449	11.9206	48.8563

TABLE I: Error and simulation time vs time horizon in receding horizon CEC controller

$\lvert\mathcal{U}\rvert$	$\mathbf{e}_{total}$	$\mathbf{e}_{\mu}$	$\mathbf{t}_{sim,Total}$(s)	$\mathbf{t}_{sim,\mu}$(ms)
6	67.4996	0.1673	36.5494	152.2928
25	63.5331	0.2383	502.6004	2093.4375
50	74.3602	0.1704	1781.4320	7421.8092

TABLE II: Error and simulation time vs size of control space$|\mathcal{U}|$ in GPI controller

B. Receding horizon CEC algorithm

Following are the parameter values used in the code implementation,

$$\begin{aligned} \mathbf{Q} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ q &= 1 \\ \mathbf{u} &= \begin{bmatrix} 1 & 2 \end{bmatrix} \\ T &= 5 \\ \gamma &= 0.8 \end{aligned}$$

Terminal code is same as stage cost such that,

$$q(\mathbf{e}) = l(\mathbf{e}, [\mathbf{0}, \mathbf{0}])$$

C. GPI algorithm

Following parameters are used in the code implementation,

$$\begin{aligned} \mathbf{Q} &= \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \\ q &= 2 \\ \mathbf{u} &= \begin{bmatrix} 0.2 & 0.2 \end{bmatrix} \\ T &= 3 \\ dx &= 0.1 \\ dy &= 0.1 \\ d_\theta &= \frac{\pi}{10} \\ \gamma &= 0.3 \end{aligned}$$

The γ parameter is selected such that γ^T for any $T > 3$ is insignificant, hence the problem approximates to an infinite horizon policy iteration. Moreover, the terminal cost is not implemented in this code.

D. Inference

- In CEC based controller, three experiments were done at time horizon 1,3 and 5. It was observed that the controller with $T = 5$ had the best performance overall.
- Moreover, the controller was sucessfully able to track the trajectory and avoid the obstacle at the same time. Controller with $T = 3$ was also able to track the trajectory but couldn't avoid the obstacle.

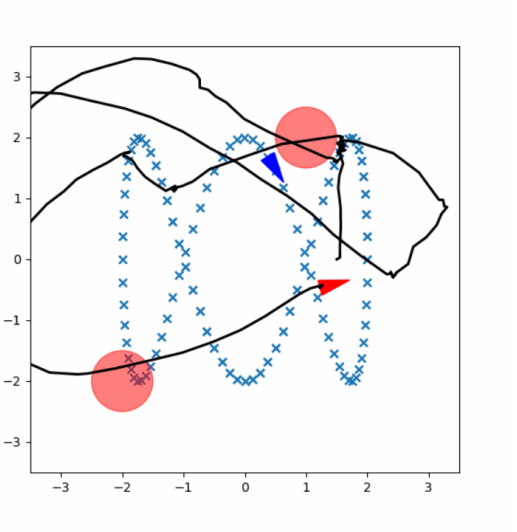

Fig. 1: Motion plan followed by the agent using receding horizon CEC algorithm. T = 1

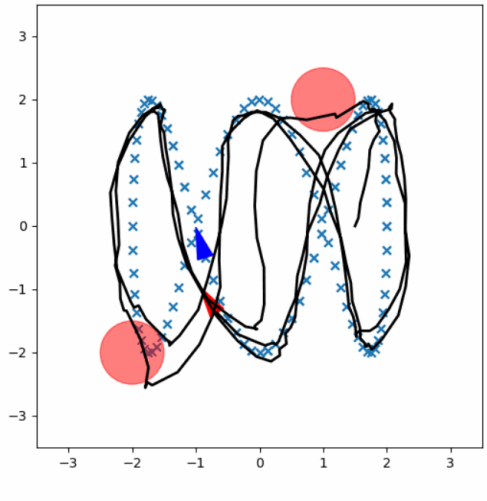

Fig. 2: Motion plan followed by the agent using receding horizon CEC algorithm. T = 3

- It was also observed that with increasing number of variable, i.e., increasing T, increased the computational complexity of the controller.
- Since there was no saddle point observed in terms of T v/s computational complexity, it is safe to say that increasing T would only increase the performance.
- GPI controllers had overall better performance(less tracking error) compared to CEC controller but were computationally intensive.
- The computation time of the controller scaled at the $\mathcal{O}(n^3)$ theoretically and the same is observed in the results.
- Although GPI had better performance in terms of tracking error it had poor performance in obstacle avoidance and

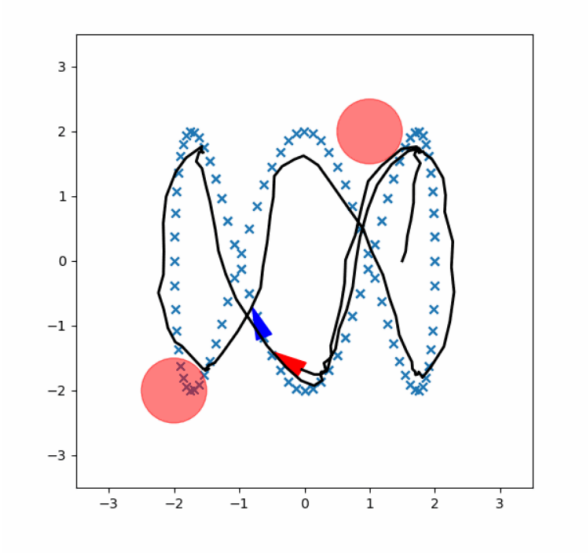

Fig. 3: Motion plan followed by the agent using receding horizon CEC algorithm. T=5

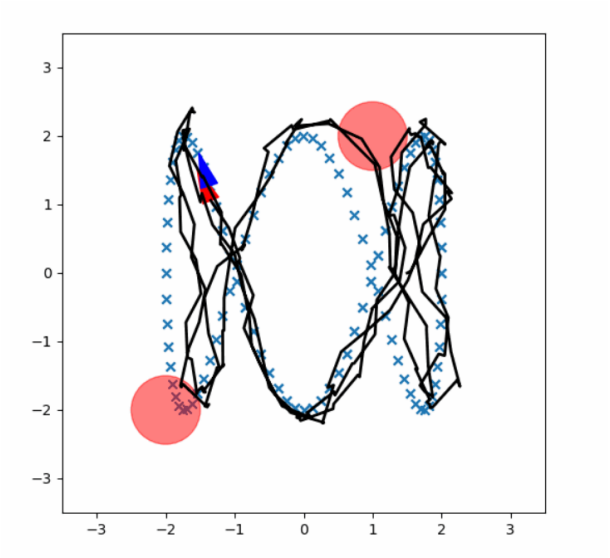

Fig. 4: Motion plan followed by the agent using GPI algorithm. $T = 3$ and $|\mathcal{U}| = 6$

it was not able to meet the objective.

- GPI controller was only implemented on coarse state space because of its computational complexity and it is fair to say that although it had better tracking, the CEC controller met both the given criteria, therefore it is the best controller among the two.
- GPI controller in the given system with very small noise standard deviation is an overkill. Instead, when there is large noise in the measurement, the CEC controller can be adapted with multiple current state estimate variables in NLP formulation.

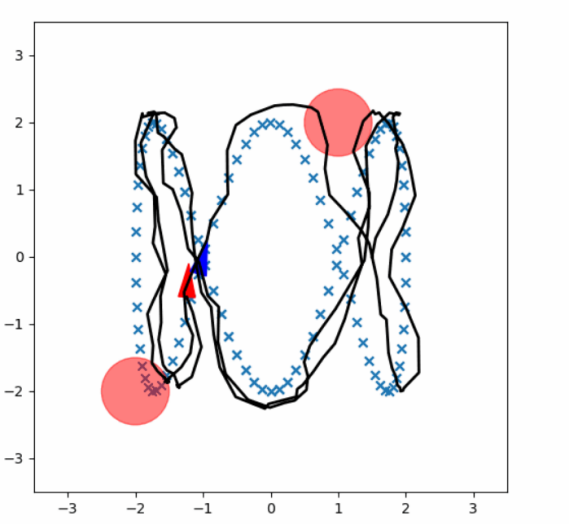

Fig. 5: Motion plan followed by the agent using GPI algorithm. $T = 3$ and $|\mathcal{U}| = 25$

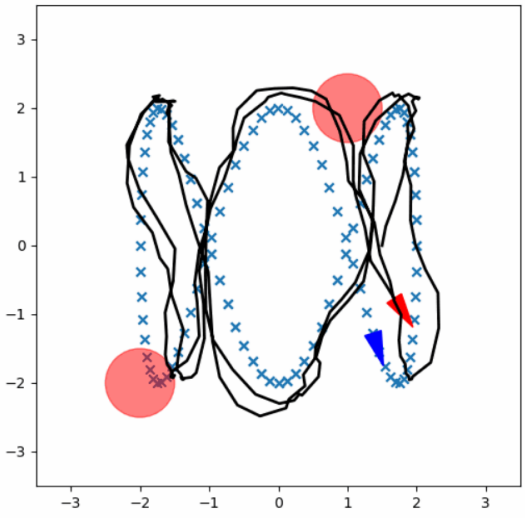

Fig. 6: Motion plan followed by the agent using GPI algorithm. $T = 3$ and $|\mathcal{U}| = 50$