Infinite-Horizon Stochastic Optimal Control

Nathan Cusson-Nadeau

I. Introduction

In robotics, it is often desirable for an agent to track a given trajectory while simultaneously avoiding any obstacles that may coincide with said trajectory. This could arise in situations where a robot is given some directions to follow and must then plan a path to achieve those directions as best as possible given some constraints of what are allowable actions. For instance, we may instruct a robot to follow some path in a warehouse that leads the robot through places without high risk of human intervention. However, this path may not always be obstacle free and the robot must avoid things like boxes or people blocking the desired trajectory. This is where optimal control can come into play.

Using infinite-horizon stochastic optimal control techniques like receding-horizon certainty equivalent control (CEC) and generalized policy iteration (GPI), the optimal lowest *cost* set of actions can be obtained in real time for a robot trying to follow some set trajectory. In the following sections, these two methods will be assessed and compared when applied to a differential-drive robot tracking a trajectory while avoiding obstacles.

II. Problem Formulation

Consider a robot whose state $x_t := (\mathbf{p_t}, \theta_t)$ 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's linear $v_t \in \mathbb{R}$ and angular $\omega_t \in \mathbb{R}$ velocities are controlled by a velocity input $\mathbf{u}_t := (v_t, \omega_t)$. The discrete-time kinematics model $\mathbf{x}_{t+1}\ \forall\ t$ of the differential-drive robot obtained from Euler discretization of the continuous-time kinematics with time interval $\Delta > 0$ is:

$$f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) = \begin{bmatrix} \mathbf{p}_{t+1} \\ \theta_{t+1} \end{bmatrix} = \dots$$

$$\dots = \underbrace{\begin{bmatrix} \mathbf{p}_t \\ \theta_t \end{bmatrix}}_{\mathbf{x}_t} + \underbrace{\begin{bmatrix} \Delta\cos\theta_t & 0 \\ \Delta\sin\theta_t & 0 \\ 0 & \Delta \end{bmatrix}}_{\mathbf{G}(\mathbf{x}_t)} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u}_t} + \mathbf{w}_t \quad (1)$$

$$t = 0, 1, 2 \dots$$

here $\mathbf{w}_t \in \mathbb{R}^3$ models the motion noise with Gaussian distribution $\mathcal{N}(\mathbf{0}, \text{diag}(\sigma)^2$ with standard deviation $\sigma = [0.04, 0.04, 0.004] \in \mathbb{R}^3$. The motion noise is assumed to be independent across time and of the robot state $\mathbf{x}_t$. The kinematics model in (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 $\mathbf{x}_t + G(\mathbf{x}_t)\mathbf{u}_t$ and covariance $\text{diag}(\sigma)^2$. The control input $\mathbf{u}_t$ of the vehicle is limited to an allowable set of linear and angular velocities $\mathcal{U} := [0, 1] \times [1, 1]$.

The objective is to design a control policy $\pi(\mathbf{x}_t)$ 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 $r_1 = 0.5$ and $\mathcal{C}_2$ centered at $(1, 2)$ with radius $r_2 = 0.5$. 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 Figure 1.

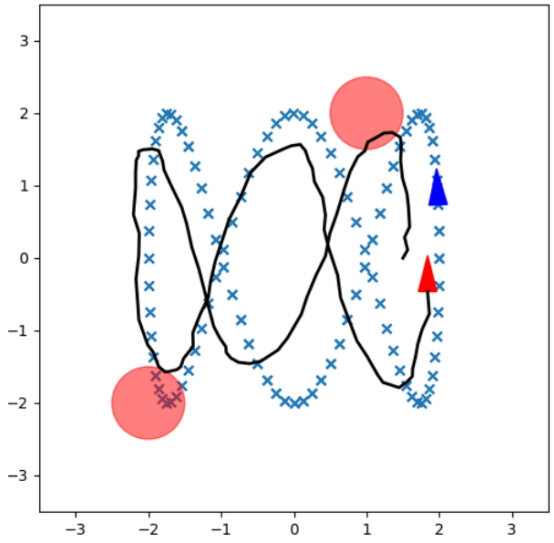

Fig. 1: Environment: The differential-drive robot (red triangle) tracks a reference lissajous curve trajectory (cyan crosses), while avoid the obstacles (red circles). The black line is the trajectory of the differential-drive robot in time produced by a proportional controller.

We define an error state $\mathbf{e}_t := (\tilde{\mathbf{p}}_t, \tilde{\theta}_t)$, where $\tilde{\mathbf{p}} := \mathbf{p}_t - \mathbf{r}_t$ and $\tilde{\theta}_t := \theta_t - \alpha_t$ measure the position and orientation deviation from the reference trajectory, respectively. The equations of motion for the error dynamics are:

$$g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t) = \underbrace{\begin{bmatrix} \tilde{\mathbf{p}}_t \\ \tilde{\theta}_t \end{bmatrix}}_{\mathbf{e}_t} + \underbrace{\begin{bmatrix} \Delta\cos\tilde{\theta}_t + \alpha_t & 0 \\ \Delta\sin\theta_t + \alpha_t & 0 \\ 0 & \Delta \end{bmatrix}}_{\tilde{\mathbf{G}}(\mathbf{e_t})} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u}_t} + \quad (2)$$

$$\dots + \begin{bmatrix} \mathbf{r}_t - \mathbf{r}_{t+1} \\ \alpha_t + \alpha_{t+1} \end{bmatrix} + \mathbf{w}_t$$

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

$$
\begin{aligned}
V^*(\tau, \mathbf{e}) = & \min_{\mathbf{u}_\tau, \ldots, \mathbf{u}_{\tau+T-1}} \mathfrak{q}(\mathbf{e}_{\tau+T}) + \ldots \\
& \cdots + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau} \left(\tilde{\mathbf{p}}_t^\top Q \tilde{\mathbf{p}}_t + q(1 - \cos(\tilde{\theta}_t))^2 + \mathbf{u}_t^\top R \mathbf{u}_t \right) \\
& \text{s.t.} \quad \mathbf{e}_{t+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0}), \quad t = \tau, \ldots, \tau + T - 1 \\
& \quad \mathbf{u}_t \in \mathcal{U} \\
& \quad \tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F}
\end{aligned}
\tag{3}
$$

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

To solve this infinite-horizon stochastic optimal control problem we utilize two different approaches: *receding-horizon certainty equivalent control* (CEC) and *generalized policy iteration* (GPI).

A. Receding-Horizon Certainty Equivalent Control (CEC)

CEC applies, at each stage, a control that would be considered optimal if the noise variables $\mathbf{w}_t$ were fixed at their expected values (zeros). In using CEC, the *stochastic* optimal control problem is reduced to a *deterministic* optimal control problem. This allows the problem to be solved more efficiently. Receding-horizon CEC, additionally, approximates an infinite-horizon problem by iteratively solving the *discounted finite-horizon deterministic optimal control problem* at every time step:

$$
\begin{aligned}
V^*(\tau, \mathbf{e}) = & \min_{\mathbf{u}_\tau, \ldots, \mathbf{u}_{\tau+T-1}} \mathfrak{q}(\mathbf{e}_{\tau+T}) + \ldots \\
& \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau} \left(\tilde{\mathbf{p}}_t^\top Q \tilde{\mathbf{p}}_t + q(1 - \cos(\tilde{\theta}_t))^2 + \mathbf{u}_t^\top R \mathbf{u}_t \right) \\
& \text{s.t.} \quad \mathbf{e}_{t+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0}), \quad t = \tau, \ldots, \tau + T - 1 \\
& \quad \mathbf{u}_t \in \mathcal{U} \\
& \quad \tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F}
\end{aligned}
\tag{4}
$$

where $\mathbf{q}(\mathbf{e})$ is a suitably chosen terminal cost. The receding-horizon CEC problem is now a non-linear program (NLP) of the form:

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

where $\mathbf{U}$ and $\mathbf{E}$ represent:

CEC Constraints		
Constraints	**Lower Bound**	**Upper Bound**
$\mathbf{x}$	-3	3
$\mathbf{y}$	-3	3
α	$-\pi$	$\pi - 0.00001$
$(x-1)^2 + (y-2)^2$	0.25	∞
$(x+2)^2 + (y+2)^2$	0.25	∞
ω	-1	1
v	0	1
e_0	e_t	e_t
$\mathbf{e}_{\tau+1}$	$g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0})$	$g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0})$

TABLE I: CEC Constraints

$$
\mathbf{U} := [\mathbf{u}_\tau^\top, \ldots, \mathbf{u}_{\tau+T-1}^\top]^\top
$$

$$
\mathbf{E} := [\mathbf{e}_\tau^\top, \ldots, \mathbf{e}_{\tau+T-1}^\top]^\top
$$

Using an NLP solver with the proper constraints, (5) can be solved at every time step and generate the next control action $\mathbf{u}_t^\top$ that incurs the minimum amount of cost $c(\mathbf{U}, \mathbf{E})$. To do so, we utilized a NLP solver package for Python called CasADi. Table I presents the constraints enforced for the solver.

Most of these bounds are trivially determined from the problem statement, but the non-trivial will be discussed here. α, must range from $[-\pi, \pi)$ but to implement the open upper bound we just get extremely close to π by subtracting 0.00001, as it is impossible to get infinitesimally close on a computer. Next the free space constraint $\mathcal{F}$ was expressed by only allowing x and y values which are outside of the two circular obstacles $\mathcal{C}_1$ and $\mathcal{C}_2$. We enforce this constraint by looking at the equation of a circle and making it less than r^2 (0.25). The upper-bound is arbitrary as the x and y constraint already cut off values which could be infeasible. The initial error state e_0 is constrained to be the current error e_t so that the NLP solver can function properly. Lastly, the next error state $\mathbf{e}_{\tau+1}$ must be set to the error state resulting from the motion model (2).

To tune the performance of CEC, various hyperparameters could be adjusted. These being: the time horizon T, the control weight Q, the performance weight R, the terminal cost $\mathfrak{q}$, the orientation deviation cost q, the discount factor γ, and lastly the obstacle buffer.

Algorithm 1 walks through the CEC solution to our problem.

Algorithm 1
Receding-Horizon Certainty Equivalent Control (CEC)

1: Initialize T, Q, R, $\mathfrak{q}$, γ, $\mathfrak{q}$
2: $V_0^* \leftarrow \mathfrak{q}$
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: $V^*(\tau, \mathbf{e}) = \min_{\mathbf{u}_\tau \mathbf{u}_{\tau+T-1}} \mathfrak{q}(\mathbf{e}_{\tau+T}) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau} \left(\tilde{\mathbf{p}}_t^\top Q \tilde{\mathbf{p}}_t + q(1 - \cos(\tilde{\theta}_t))^2 + \mathbf{u}_t^\top R \mathbf{u}_t \right)$
5: Solve (4) with NLP solver
6: **return** $u^*(\mathbf{x})_k$
7: **end for**

Trials with CEC									
Trial	T	Q	R	q	q	γ	Buffer	**Error**	**Time***
1	75	2.5	5	0	10	**0.75**	0.05	1006	1631
2	75	2.5	5	0	10	**0.99**	0.05	189	454
3	**25**	**5**	8	0	10	0.99	0.05	225	180
4	**50**	5	8	0	10	0.99	0.05	186	350
5	50	5	8	**10**	10	0.99	0.05	186	362
6	50	5	8	0	**5**	0.99	0.05	189	346
7	50	5	**10**	0	10	0.99	0.05	193	391
8	50	5	**20**	0	10	0.99	0.05	223	340
9	50	3.5	8	0	10	0.99	0.05	195	384
10	75	3.5	8	0	10	0.99	0.055	186	426

TABLE II: CEC Trial Variables and Results. *Average time per iteration in milliseconds

B. Generalized Policy Iteration (GPI)

Using GPI we can solve (3), the discounted infinite-horizon stochastic optimal control problem, exactly by discretizing the state and control spaces.

At each time step, we applied a Policy Iteration, seen in algorithm 2.

ALGORITHM 2
General Policy Iteration - Using PI

1: Initialize V_0
2: **for** $k = 0, 1, 2, ...$ **do**
3: Policy Improvement:
4: $\pi_{k+1}(x) = \underset{u \in U(x)}{argmin}\ H[\mathbf{x}, \mathbf{u}, V_k(\cdot)]$
5: Policy Evaluation:
6: $V_{k+1} = \mathcal{B}^{\infty}_{pi_{k+1}}[V_k]$
7: **end for**
8: **return** $\pi^*(\mathbf{x})$

$$\begin{aligned}\mathcal{B}_\pi[V](x) :=& H[\mathbf{x}, \pi(\mathbf{x}), V] = \dots \\ & \ell(\mathbf{x}, \pi(\mathbf{x})) + \gamma \mathbb{E}_{\mathbf{x}' \sim p_f(\cdot|\mathbf{x}, \pi(\mathbf{x}))}[V(\mathbf{x}')]\end{aligned} \tag{6}$$

IV. RESULTS

A. CEC Results and Discussion

Using CEC proved quite suboptimal as an optimal control scheme when applied to this problem. 10 trials were conducted where various hyperparameters were tuned to see which would yield the best trajectories (see Table II), but none used ended in low error. Through further tuning, these trajectories may be become more optimal but regardless of what's used CEC will succumb to a pitfall. This pitfall is the ignorance of noise at every time step. While this makes the problem easier and possible to solve using a NLP, it does so at the expense of accuracy.

While CEC did not track the trajectories incredibly well, the obstacles were consistently avoided with the computed trajectories.

Figures depicting the trials from Table II can be found in the Appendix. Animated gifs illustrating the trajectories for 240 iterations can be found in the associated gifs folder in the code submission.

From analyzing these figures and recorded data, one can see that the trials (e.g. Trials 4, 5, 6, 10) which performed best (low error and low time to compute) utilized a γ near 1, a moderately large horizon time T, with a ratio of the control weight to performance weight (Q/R) of about 1:2. Terminal cost seemed to have little to no effect on the error of the final solution but did make the problem unlsolveable at larger values. Larger buffers also made the problem infeasible and was very sensitive. A larger cost of deviation q helped performance a little bit but also quickly made the problem infeasible.

One key observation is that the time horizon T improves error up to a threshold but beyond that threshold it only increases computation time. This makes sense as major trajectory changes are only needed at every turn which are encompassed in about every 20 iterations.

Another interesting observation as more clearly seen in the accompanying gifs, is the frequent spin outs the tracking agent would make at the apex of each lissajous curve. A freeze frame of this action can be seen in Figure 2. This while surprising at first, makes sense because of the sharp turns allowable by a differential-drive robot's motion model. What does not make much sense is the direction in which the robot begins spinning. It seems that in many circumstances pivot in a motion that would incur extra cost, yet CEC has found this trajectory to be optimal. Further investigation would be required to determine why this is the case but it seems probable that tuning the deviation from the proper orientation could prevent this sort of behavior.

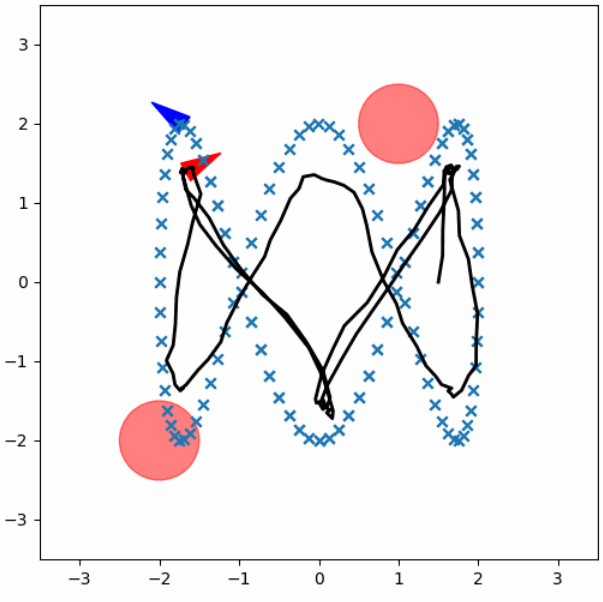

Fig. 2: Trial 2: Trajectory

Given more time, CEC could likely produce excellent tracking performance but more tuning with parameters would

be necessary. Tinkering with the parameters longer could give a better feel for which direction to look. As of now, it seems that playing with the values of Q and R would have the most influence on the tracking and this would be the first fine tuning that would be attempted to improve performance.

B. GPI Results and Discussion

Personal time-constraints did not allow for GPI to be completed in time. We can expect GPI to have better performance due to it actually taking the noise into account in it's computation

V. Appendix

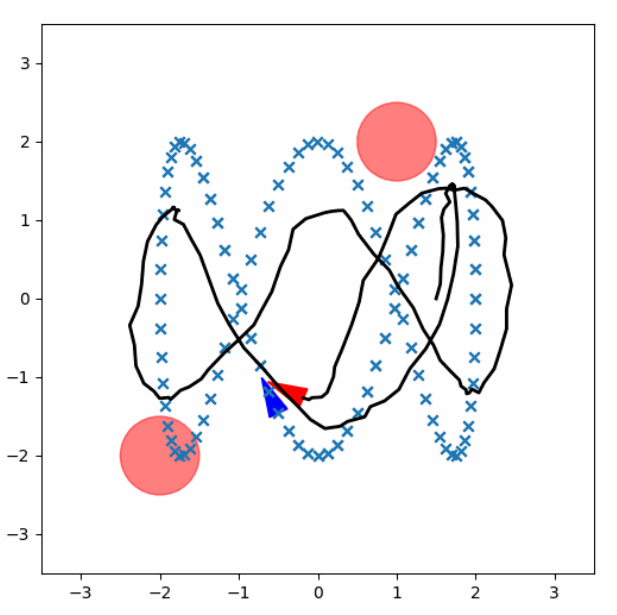

Fig. 3: Trial 1: Trajectory

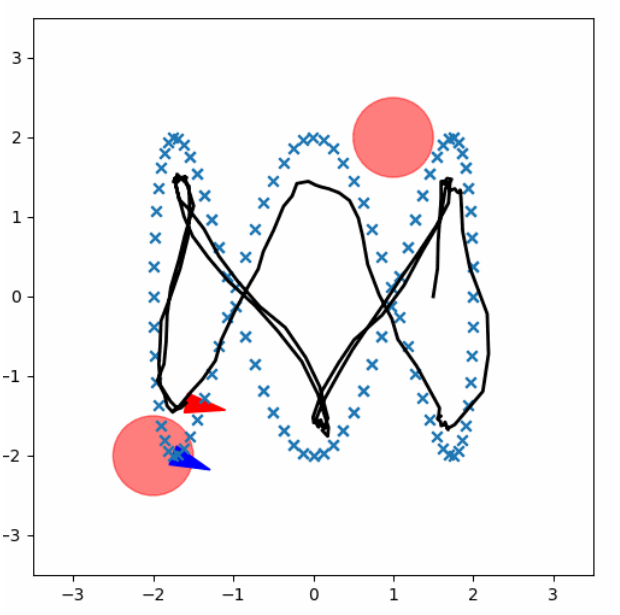

Fig. 4: Trial 3: Trajectory

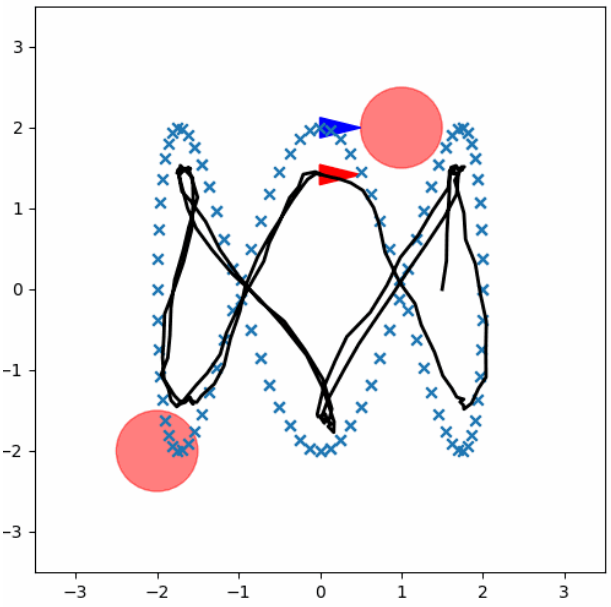

Fig. 5: Trial 4: Trajectory

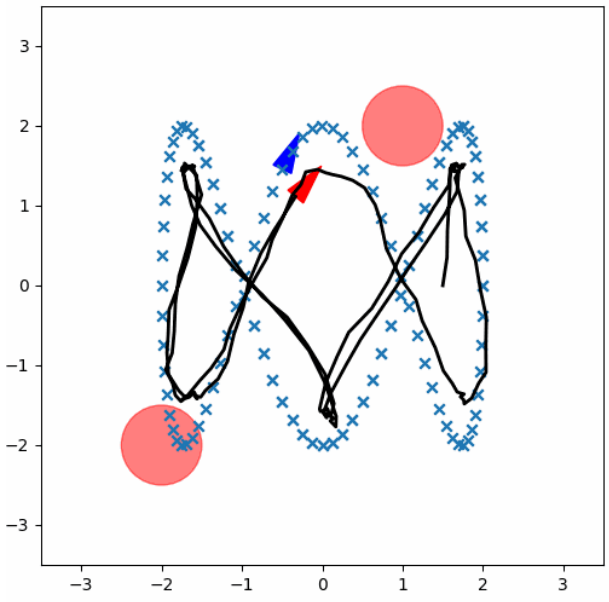

Fig. 6: Trial 5: Trajectory

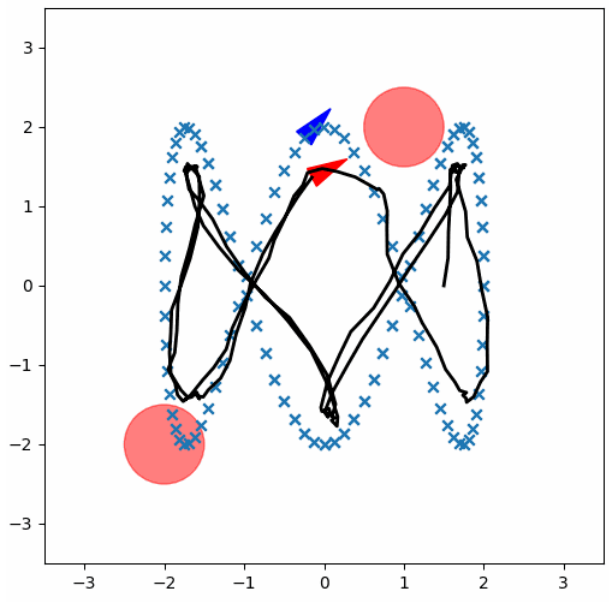

Fig. 7: Trial 6: Trajectory

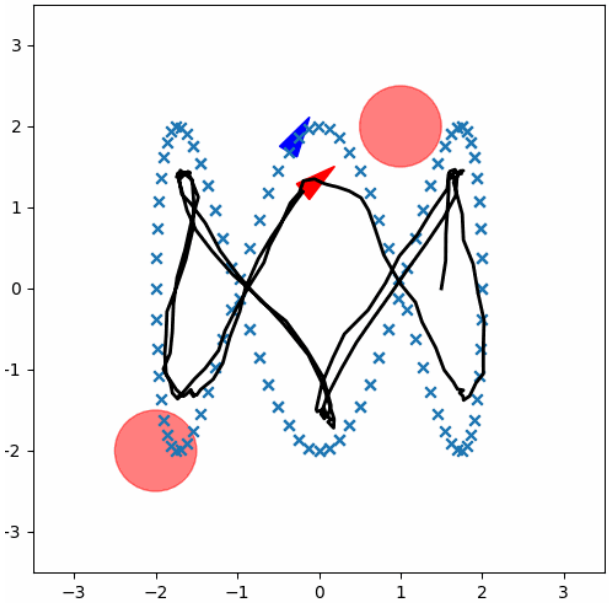

Fig. 8: Trial 7: Trajectory

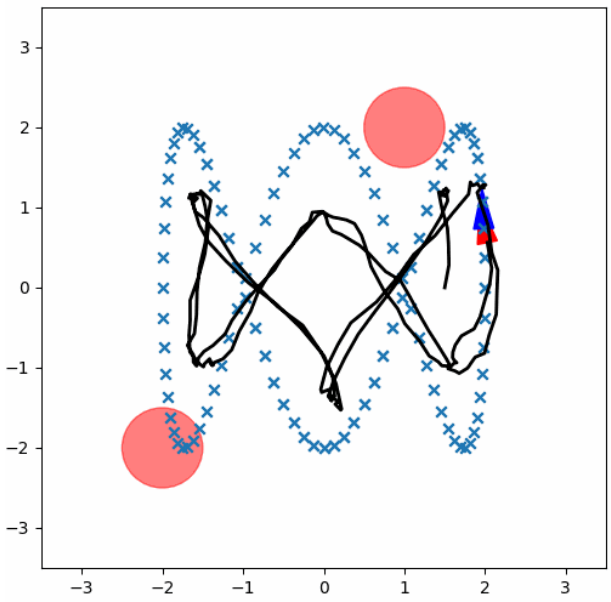

Fig. 9: Trial 8: Trajectory

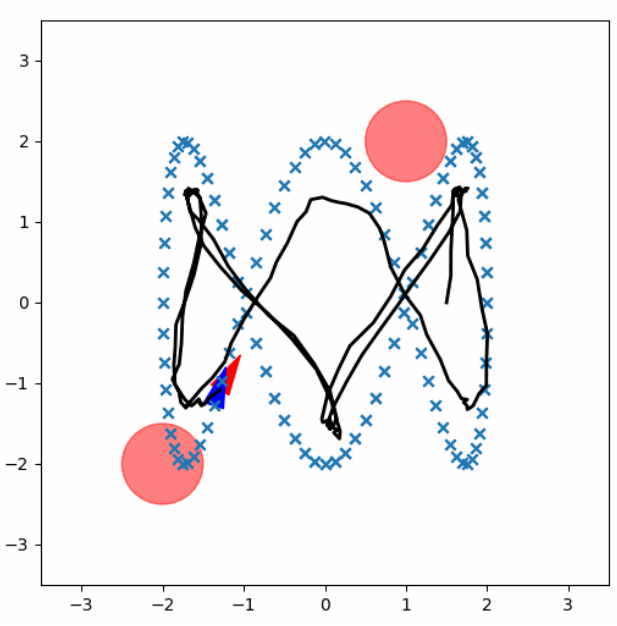

Fig. 10: Trial 9: Trajectory

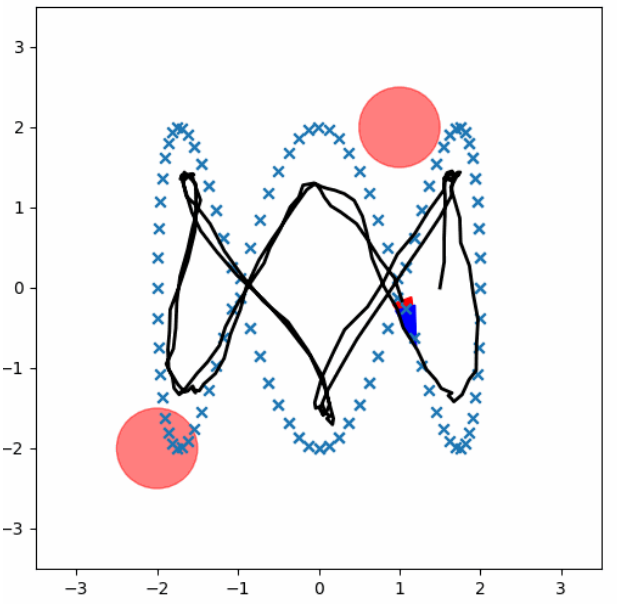

Fig. 11: Trial 10: Trajectory