Infinite-Horizon Stochastic Optimal Control

Hsiu-Wen Yen A59010599 hsyen@ucsd.edu

I. Introduction

Stochastic optimal control is a method to design control policy under uncertainty of motion model. It's very important because noise is always existed in reality no matter what equipment we are using. Using stochastic optimal control makes simulation more realistic.

In this problem, we have a robot on 2d coordinate system and a desired trajectory which represents what current robot's position should be at any timestamp. While tracking the trajectory, our robot should also avoid obstacle on the path. We can actuate our robot by controlling its linear velocity and angular velocity. However, our control input will be corrupted by noises.

To find the optimal control under uncertainty of noise, we use two approaches to solve the problem. One is certainty equivalent control (CEC), which reduces the stochastic optimal control problem to a deterministic one. Another is general policy iteration (GPI), which perform policy iteration with finite time in computing policy evaluation.

II. Problem Formulation

In this problem, our robot is a differential-drive robot which has state $x(t) = (p(t), \theta(t))$, and it is controlled by linear velocity and angular velocity $u(t) = (v(t), \omega(t))$.

$$x(t) = \begin{bmatrix} p(t) \\ \theta(t) \end{bmatrix} \tag{1}$$

$$p(t) \in R^2 \tag{2}$$

$$\theta(t) \in [-\pi, \pi) \tag{3}$$

Computer can't compute continuous time system, so we use Euler discretization with time interval Δ to make it a discrete time model.

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

$$\Delta > 0$$

$$t = 0,1,2,\ldots \tag{4}$$

However, our motion model will be influenced by noise w. Therefore, there will be adding noise term to the next state.

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

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

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

$$w_t \in R^3 \tag{5}$$

Let w follow an independent zero-mean Gaussian distribution, we can define the probability density function for motion model.

$$pf(x_{t+1}|x_t, u_t) \sim N(\mu, \Sigma) \tag{6}$$

$$\mu = \begin{bmatrix} p_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta\cos(\theta_t) & 0 \\ \Delta\sin(\theta_t) & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \tag{7}$$

$$\Sigma = \begin{bmatrix} {\sigma_1}^2 & 0 & 0 \\ 0 & {\sigma_2}^2 & 0 \\ 0 & 0 & {\sigma_3}^2 \end{bmatrix} \tag{8}$$

We can also discretize the desired trajectory as reference position trajectory r_t and orientation trajectory.

$$r_t \in R^2$$

$$\alpha_t \in [-\pi, \pi)$$

$$t = 0,1,2,\ldots \tag{9}$$

The desired trajectory is fixed, so we can redefine an error state e_t as state space.

$$e_t = (\widetilde{p_t}, \widetilde{\theta_t}) \tag{10}$$

$$\widetilde{p_t} = p_t - r_t \tag{11}$$

$$\widetilde{\theta_t} = \theta_t - \alpha_t \tag{12}$$

The equations of motion for the error dynamics are

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

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

$$t = 0,1,2,\ldots \tag{13}$$

There are obstacles in the environment. Therefore, we need to define the free space F in $[-3,3]^2$. In this problem, we have two obstacles. One is C_1 centered at $(-2,-2)$ with radius 0.5, another is C_2 centered at (1,2) with radius 0.5. Then the free space is

$$F = [-3,3]^2 \setminus (C_1 \cup C_2) \tag{14}$$

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

$$V^*(\tau, e) =$$
$$min_\pi \mathbb{E}\left[\sum_{t=\tau}^{\infty} \gamma^{t-\tau}\left(\tilde{p}_t{}^T Q \tilde{p}_t + q\left(1-\cos(\tilde{\theta}_t)\right)^2 + u_t^T R u_t\right) | e_\tau = e\right]$$
$$e_{t+1} = g(t, e_t, u_t, w_t),\ w_t \sim N(0, diag(\sigma)^2),\ t = \tau, \tau+1, \dots$$
$$u_t = \pi(t, e_t) \in U$$
$$\gamma = 0 \sim 1$$
$$\tilde{p}_t + r_t \in F \tag{15}$$

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

III. THECTIVAL APPROCHES

A. CEC

If the noise variables w_t were fixed at their expected values, we can use CEC to reduce a stochastic optimal control problem to a deterministic optimal control problem.

Then, we use receding-horizon CEC to approximate the infinite-horizon problem to a discounted finite-horizon deterministic optimal control problem.

$$V^*(\tau, e_\tau) =$$
$$min_\pi\ \boldsymbol{q}(e_{\tau+T}) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau}\left(\tilde{p}_t{}^T Q \tilde{p}_t + q\left(1-\cos(\tilde{\theta}_t)\right)^2 + u_t^T R u_t\right)$$
$$e_{t+1} = g(t, e_t, u_t, 0),\ t = \tau, \dots, \tau+T-1$$
$$\pi = u_\tau, \dots, u_{\tau+T-1}$$
$$u_t \in U$$
$$\gamma = 0 \sim 1$$
$$\tilde{p}_t + r_t \in F \tag{16}$$

$\boldsymbol{q}(e)$ is a suitably chosen terminal cost.

We can treat (16) as a non-linear program (NLP). By minimizing the objective function over the optimization variable under constraints, we can get the optimal control policy at time τ.

To solve NLP, we need to define the objective function, optimization variables, and constraints.

Objective function $obj(\tau, e_\tau)$:

$$\boldsymbol{q}(e_{\tau+T}) + \sum_{t=\tau}^{\tau+T-1} \gamma^{t-\tau}\left(\tilde{p}_t{}^T Q \tilde{p}_t + q\left(1-\cos(\tilde{\theta}_t)\right)^2 + u_t^T R u_t\right) \tag{17}$$

First, we need to define a proper terminal cost $\boldsymbol{q}(e)$, and I choose $\boldsymbol{q}(e) = 0$. This is because we will recompute the control policy for every timestamp, so ignoring the terminal cost won't be a big problem and makes the function simple.

Second, I change the terminal time T for any timestamp τ to T^*, where $T^* = \min(T, \tau + k)$. k is a hyperparameter.

$$T^* = \min(T, \tau + k) \tag{18}$$
$$k \ge 1 \tag{19}$$

This is a suitable method to speed up computation speed because the exponential term of γ makes latter terms insignificant to our objective function.

Therefore, we can rewrite the objective function as

$$\sum_{t=\tau}^{\tau+T^*-1} \gamma^{t-\tau}\left(\tilde{p}_t{}^T Q \tilde{p}_t + q\left(1-\cos(\tilde{\theta}_t)\right)^2 + u_t^T R u_t\right) \tag{20}$$

Optimization variables:

$$u_\tau, \dots, u_{\tau+T^*-1}$$

*Only e_τ is provided by input argument. $e_{\tau+1}, \dots, e_{\tau+T^*}$ is obtained by (13) which depends on $u_\tau, \dots, u_{\tau+T^*-1}$.

Constraints:

1. Free space constraints: $\tilde{p}_t + r_t \in F$

Our robot can't move outside the map, so there are boundary constraints:

$$\tilde{p}_t + r_t \in [-3,3]^2, \forall t = \tau, \dots, \tau + T^* \tag{21}$$

Our robot can't collide with obstacles, so there are collision constraints:

$$||\tilde{p}_t + r_t - c_i||_2 \ge r_i, \forall t = \tau, \dots, \tau + T^* \tag{22}$$

c_i is center of circular obstacle C_i with radius r_i.

2. Control constraints: $\pi(\tau, e_\tau) = u_\tau, \dots, u_{\tau+T^*-1} \in U$

$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}$$
$$0 \le v_t \le 1$$
$$-1 \le \omega_t \le 1$$
$$t = \tau, \dots, \tau + T^* - 1 \tag{23}$$

By solving NLP above, we can get $u_\tau, \dots, u_{\tau+T^*-1}$ at given timestamp τ and error state e_τ. Then, we let robot make a move with u_τ. Because of noise it will reach a new error state $\overline{e_{\tau+1}}$, now we can change input to $(\tau + 1, \overline{e_{\tau+1}})$ to recompute the optimal control policy and repeat the process until $\tau = T$.

```
Pseudo Code: CEC
$p = [x_{init}, y_{init}]$
$\theta = \theta_{init}$
$x = [p, \theta]$
for t = 0,1,2,...,T:
    $e = \begin{bmatrix} p \\ \theta \end{bmatrix} - \begin{bmatrix} r_t \\ \alpha_t \end{bmatrix}$
    NLP objective function = $obj(t, e)$
    Solve NLP to get $u_t, \dots, u_{t+T^*-1}$
    $u = u_t \leftarrow$ take first policy as input
    $\bar{x} = f(x, u, w) \leftarrow$ motion model with noise
    Plot $\bar{x} \leftarrow$ record real postion of robot
```

$p = \bar{x}[0]$
$\theta = \bar{x}[1]$
$x = \bar{x}$

B. GPI

To run policy iteration we need to discretize the state and control spaces. Discretize state and control space into $(n_t, n_x, n_y, n_\theta)$ and (n_v, n_ω) number of grid points.

$$t = \{t_1, t_2, \dots, t_{nt}\}, t_1 = 0, t_{nt} = T$$

$$x = \{x_1, x_2, \dots, x_{nx}\}, x_1 = -3, x_{nt} = 3$$

$$y = \{y_1, y_2, \dots, y_{ny}\}, y_1 = -3, y_{nt} = 3$$

$$\theta = \{\theta_1, \theta_2, \dots, \theta_{n\theta}\}, \theta_1 = -\pi, \theta_{nt} = \pi$$

$$v = \{v_1, v_2, \dots, v_{nv}\}, v_1 = 0, v_{nt} = 1$$

$$\omega = \{\omega_1, \omega_2, \dots, \omega_{n\omega}\}, \omega_1 = -1, \omega_{nt} = 1$$

For any given state $[t, x, y, \theta]$ and control $[v, \omega]$, we can build transition probabilities matrix P by (6). Where P has size $n_t \times n_x \times n_y \times n_\theta$.

However, (6) can't handle collision checking. To solve this problem, we can define an additional stage cost z to penalize collisions.

$$z(t, e) = \begin{cases} 0, if\ ||\tilde{p} + r_t - c_i||_2 \geq r_i \\ \infty, if\ ||\tilde{p} + r_t - c_i||_2 < r_i \end{cases} \tag{24}$$

Therefore, we can rewrite value function V^π for policy π for policy evaluation as

$$V^\pi(\tau, e) = \left[z(\tau, e) + \left(\tilde{p}^T Q \tilde{p} + q\left(1 - \cos(\tilde{\theta})\right)^2 + u_\tau^T R u_\tau\right)\right] + \left[\gamma \sum_{e'} pf(e' | e, \pi(\tau, e))\, V^\pi(\tau + 1, e')\right] \quad \forall \tau, e \tag{25}$$

The policy improvement is

$$\pi(\tau, e) = argmin_u [z(\tau, e) + (\tilde{p}^T Q \tilde{p} + q(1 - \cos(\tilde{\theta}))^2 + u^T R u)] + \left[\gamma \sum_{e'} pf(e' | e, u)\, V^\pi(\tau + 1, e')\right] \quad \forall u \tag{26}$$

The difference between GPI and normal policy iteration is we only run finite time in policy evaluation (don't need to keep computing until converge).

IV. Results

A. CEC

1. Find suitable k

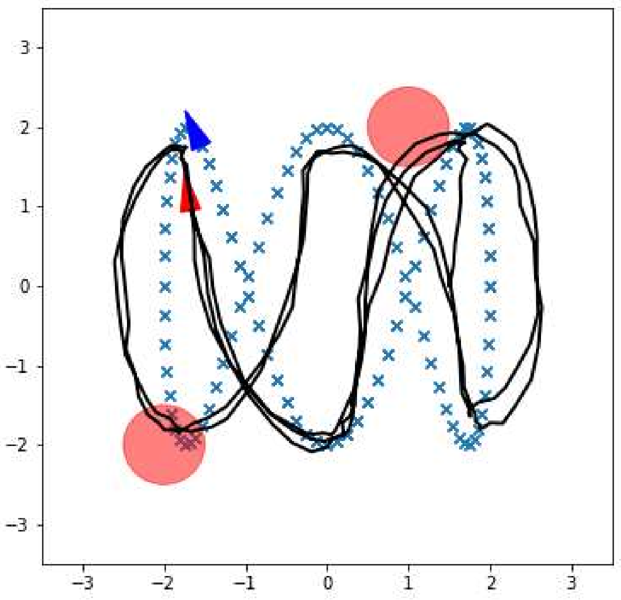
In this problem, Q, q, R, γ will be defined by a complete problem description. Therefore, the only tunable parameters here is k. Run a few experiments on different k to choose a suitable k.

Fixed parameters:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, q = 1, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \gamma = 0.85$$

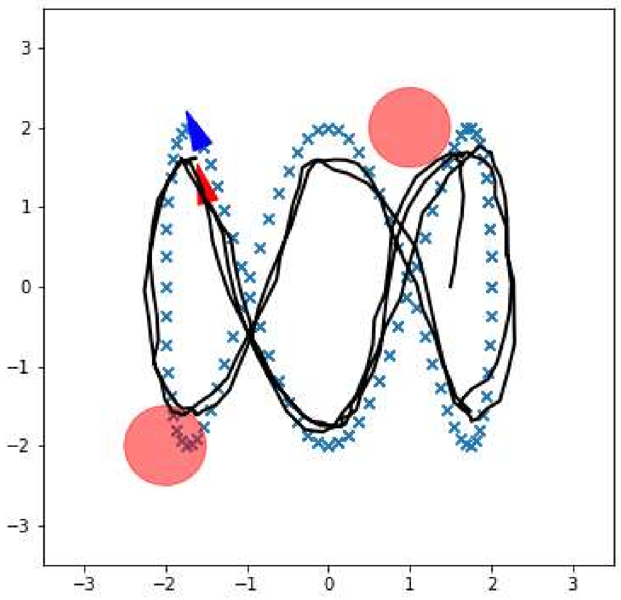
*In this part, I disable collision checking to exhibit the difference of different k more clearly.

k = 5



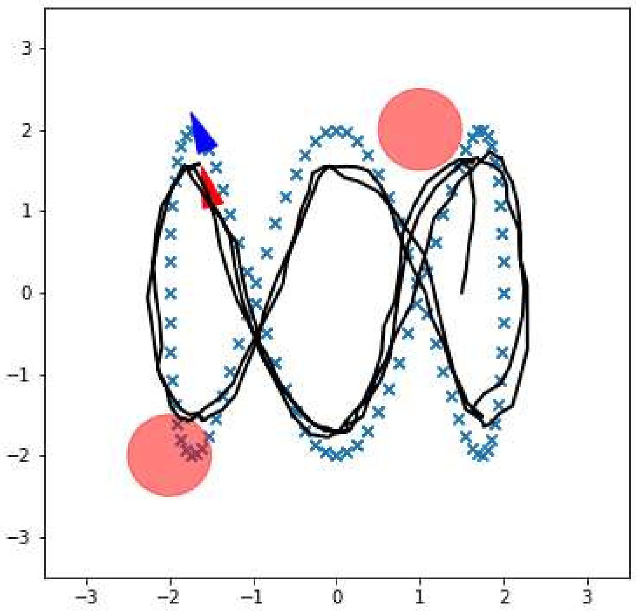
Total time: 7.57s
Average iteration time: 31.08ms
Final error: 2186.21

k=10



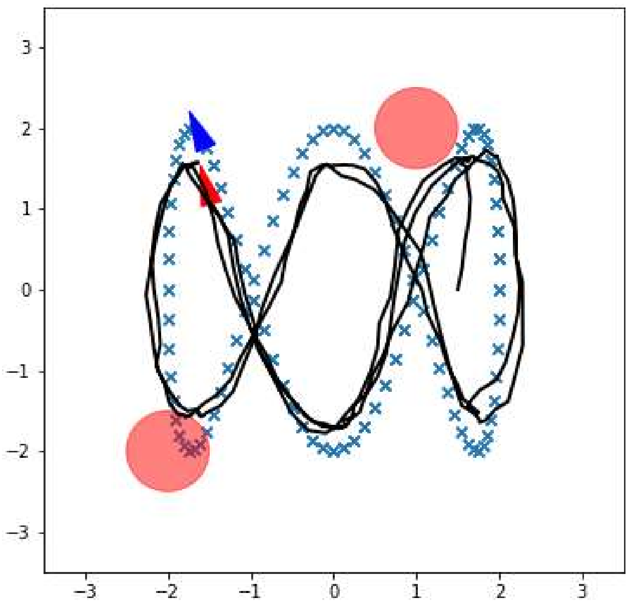
Total time: 14.34s
Average iteration time: 59.23ms
Final error: 2188.32

k=20



Total time: 41.13s
Average iteration time: 170.91ms
Final error: 2189.83

k=30



Total time: 87.41s
Average iteration time: 352.65ms
Final error: 2189.91

Discussion:
From observation above, we can see that when k increases, average iteration time increases a lot. However, the result didn't change a lot when k > 10. Therefore, we can fix k to 10 to speed up computation speed when $\gamma = 0.85$.

2. Define proper $\boldsymbol{Q}, \boldsymbol{q}, \boldsymbol{R}$
Normally, Q, q, R are defined by a complete problem description. However, in this problem we have freedom to define these parameters to reach a perfect tracking.

For simplicity, I redefine Q, R as a multiplication of identity matrix.

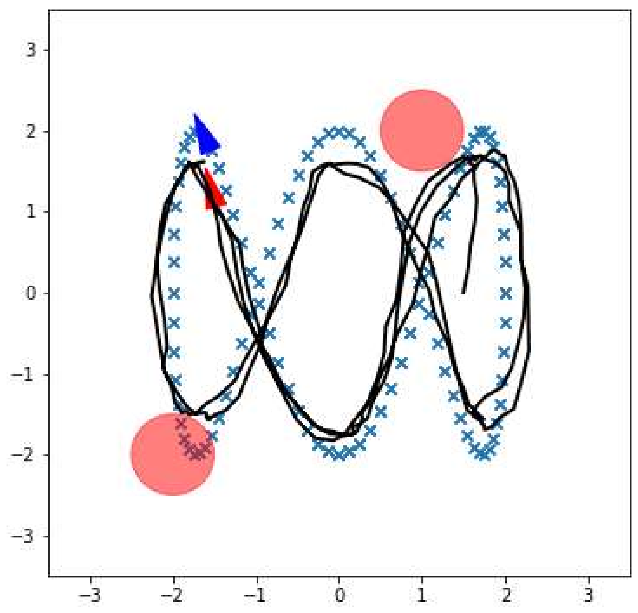
$$Q = Q * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$R = R * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Now, Q, R are scalar and easy to tune.

Fixed parameters:
$$\gamma = 0.85, k = 10$$

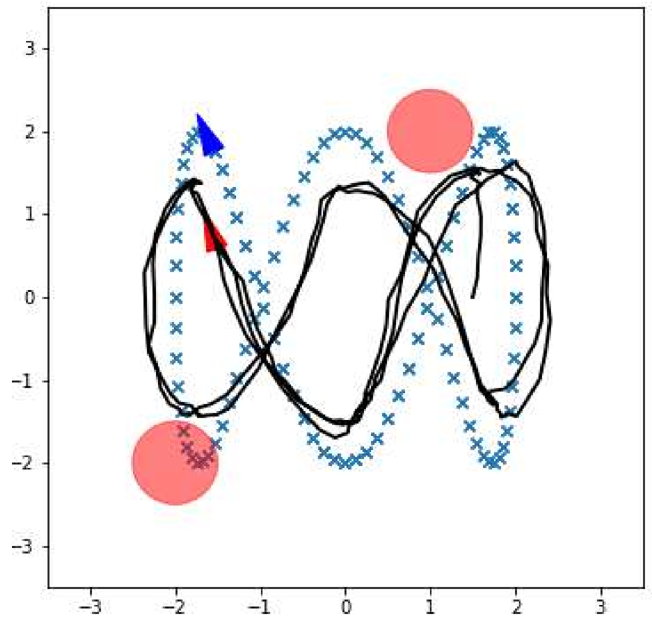
*In this part, I enable collision checking to exhibit the performance in real situation.

$Q = 1, q = 1, R = 1$



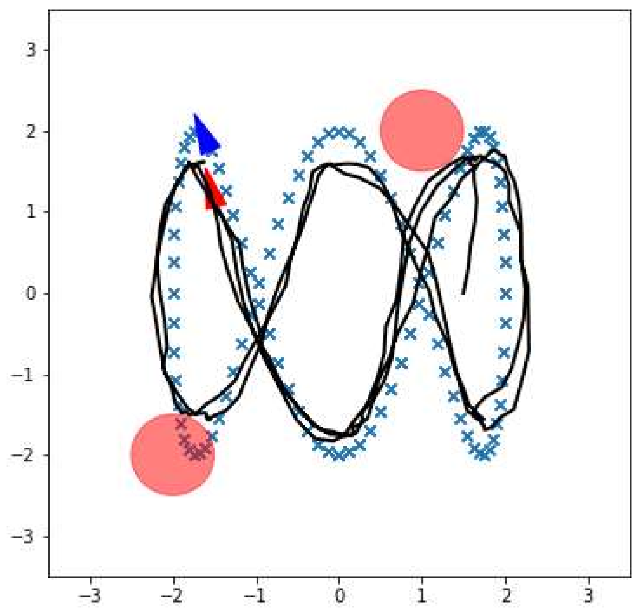
Total time: 15.84s
Average iteration time: 65.49ms
Final error: 2188.32

$Q = 0.5, q = 0.5, R = 1$



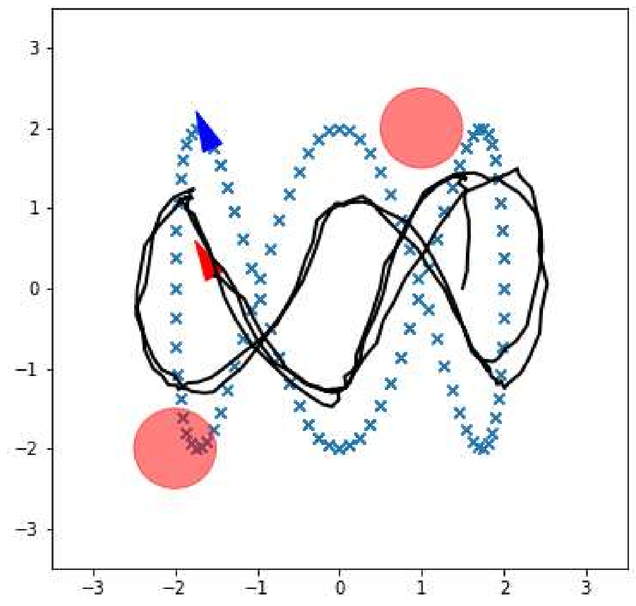
Total time: 23.15s
Average iteration time: 96.00ms
Final error: 2183.71

$Q = 1.3, q = 1.3, R = 1$



Total time: 22.30s
Average iteration time: 92.40ms
Final error: 2190.78

$Q = 1, q = 1, R = 3$



Total time: 21.76s
Average iteration time: 90.21ms
Final error: 2179.61

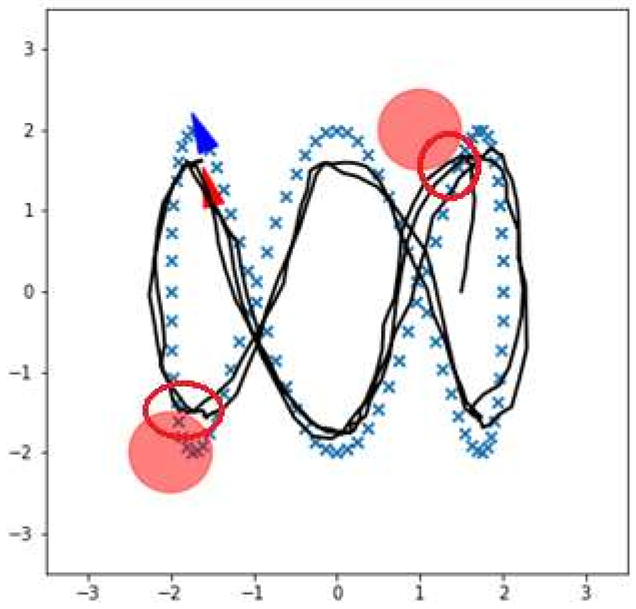
Discussion:
From observation above, we can see when $R > Q, q$, the paths deviate a lot. This is because our controller wants to reduce the energy consumption, so it has smaller movement. Overall, we can see $Q = 1, q = 1, R = 1$ has closer

trajectory and quicker average iteration time. Therefore, we can choose $Q = 1, q = 1, R = 1$ to define system.

3. Collision improvement
Even though we have collision constraints (22), we still can't guarantee that its next move won't collide with obstacles due to the noise.

For example

$Q = 1, q = 1, R = 1, k = 10, \gamma = 0.85$

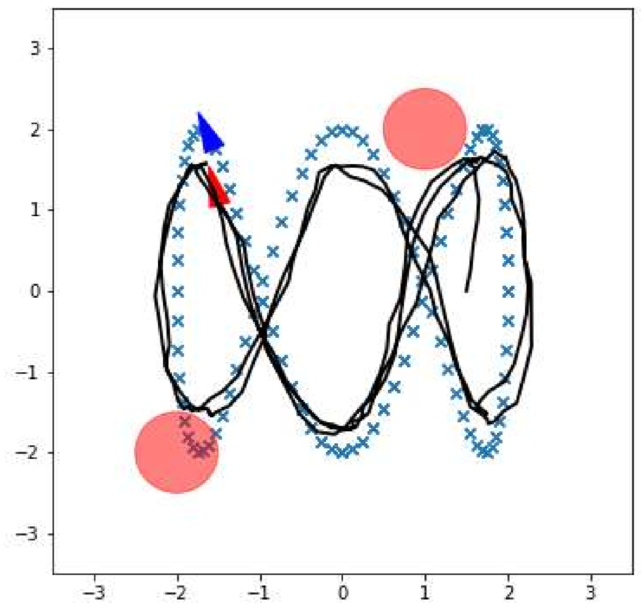


Red circles show that these two points are highly possible to collide with obstacles.

To eliminate this problem, my solution is to increase obstacles' radius in (22) as safety clearance.

In this problem, our obstacle radius is 0.5. Therefore, I set radius in collision checking to 0.52.

$Q = 1, q = 1, R = 1, k = 10, \gamma = 0.85, r_i = 0.52$



We can see that change r_i to 0.52 create more free space between obstacles. Therefore, it's a valid method to improve collision checking.

V. Reference

1. Professor Nikolay Atanasov's slides (lecture10~12)