# Comparing Receding-horizon Certainty Equivalent Control and Generalized Policy Iteration

**Allen Zeng**
UCSD
azeng@ucsd.edu

## Abstract

This report compares the performance of using a *receding-horizon certainty equivalent control* (CEC) approach and *generalized policy iteration* (GPI) approach towards solving a problem of safe trajectory tracking for a ground differential-drive robot. The robot must track a lissajous curve while avoiding obstacles in the environment. Quantitative and qualitative analysis of the two different control schemes is provided.

## 1 Introduction

Robots may be tasked to follow a target trajectory while avoiding obstacles in a continuous space, real-world environment. It may be the case that the target trajectory may collide with the given obstacles. So, it is important to devise an optimal control policy which minimizes the robotic agent's deviation from the target trajectory while also avoiding obstacles. Here, we solve this control problem in a *receding-horizon certainty equivalent control* (CEC) approach and *generalized policy iteration* (GPI) approach.

In our 2D continuous space environment, the ground differential-drive robot must follow a given lissajous curve while avoiding two circular obstacles [1]. In the CEC approach, we model the problem as a non-linear program (NLP). We assume the motion noise is fixed at the expected value of zero, and at each timestep find the control that would be optimal under a limited time horizon. Overall this control scheme is suboptimal, but is computationally efficient and gives reasonable results. In the GPI approach, we discretize the state and control spaces into a number of grid points, and use Policy Iteration [2] to find the optimal controls. Because of the discretization process, the GPI results are suboptimal in continuous space, but are reasonable as well.

## 2 Problem Formulation

We use the same problem formulation as specified in [1], and so using their notation we setup the problem as follows.

We consider a robot with state $\mathbf{x}_t = (\mathbf{p}_t.\theta_t)$ at discrete time $t \in \mathbb{N}$, where position $\mathbf{p}_t \in \mathbb{R}^2$ and orientation $\theta_t \in [-\pi, \pi)$. The robot has control inputs $\mathbf{u}_t = (v_t, \omega_t) \in R^2$ consisting of linear and angular velocity respectively. For time interval $\tau > 0$, the discrete-time kinematic model of the differential-drive robot (using Euler discretization) is:

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) = \underbrace{\begin{bmatrix} \mathbf{p}_t \\ \theta_t \end{bmatrix}}_{\mathbf{x}_t} + \underbrace{\begin{bmatrix} \tau\cos(\theta_t) & 0 \\ \tau\sin(\theta_t) & 0 \\ 0 & \tau \end{bmatrix}}_{\mathbf{G}(\mathbf{x}_t)} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u}_t} + \mathbf{w}_t, \quad t = 0, 1, 2, ... \tag{1}$$

In the above equation $\mathbf{w}_t \in \mathbb{R}^3$ models the motion noise as a Gaussian distribution $\mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\sigma})^2)$ where $\boldsymbol{\sigma} = [0.04, 0.04, 0.004] \in \mathbb{R}^3$. The motion noise is assumed to be independent of time and robot state. The above kinematics model defines the probability density function $p_f(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t$ as the density of a Gaussian distribution with mean $\mathbf{x}_t + \mathbf{G}(\mathbf{x}_t)\mathbf{u}_t$ and covariance $\text{diag}(\boldsymbol{\sigma})^2$. The control input $\mathbf{u}_t$ is limited to linear and angular velocities in $\mathcal{U} = [0, 1] \times [-1, 1]$.

The objective is to find a control policy for the robot that tracks a reference position trajectory $\mathbf{r}_t \in \mathbb{R}^2$ and orientation trajectory $\alpha_t \in [-\pi, \pi)$ while avoiding two circular obstacles: $\mathcal{C}_1$ centered at $(-2, -2)$, $\mathcal{C}_2$ centered at $(1, 2)$, both with radius 0.5. Let the free space be denoted as $\mathcal{F} = [-3, 3]^2 \backslash (\mathcal{C}_1 \cup \mathcal{C}_2)$. This environment can be seen in the results figures of Section 4. Here, we follow a lissajous curve with parameters defined by [1], which has a period of $n_t = 100$ timesteps.

For convenience, we define an error state $\mathbf{e}_t = (\tilde{\mathbf{p}}_t, \tilde{\theta}_t)$ where $\tilde{\mathbf{p}}_t = \mathbf{p}_t - \mathbf{r}_t$ and $\tilde{\theta}_t = \theta_t - \alpha_t$. This corresponds to position and orientation deviation from the reference trajectory respectively.

$$\mathbf{e}_{t+1} = \begin{bmatrix} \tilde{\mathbf{p}}_{t+1} \\ \tilde{\theta}_{t+1} \end{bmatrix} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t) = \underbrace{\begin{bmatrix} \tilde{\mathbf{p}}_t \\ \tilde{\theta}_t \end{bmatrix}}_{\mathbf{x}_t} + \underbrace{\begin{bmatrix} \tau\cos(\tilde{\theta}_t + \alpha_t) & 0 \\ \tau\sin(\tilde{\theta}_t + \alpha_t) & 0 \\ 0 & \tau \end{bmatrix}}_{\tilde{\mathbf{G}}(\mathbf{x}_t)} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\mathbf{u}_t} + \begin{bmatrix} \mathbf{r}_t - \mathbf{r}_{t+1} \\ \alpha_t - \alpha_{t+1} \end{bmatrix} + \mathbf{w}_t \quad (2)$$

Now with initial time $\tau$ and initial tracking error $\mathbf{e}$, we define the discounted infinite-horizon stochastic optimal control problem that we solve in this report:

$$\begin{aligned} V^*(\tau, \mathbf{e}) = \min_{\pi} \mathbb{E}\left[\sum_{t=\tau}^{\infty} \gamma^{(t-\tau)} \left(\tilde{\mathbf{p}}_t^\top \mathbf{Q} \tilde{\mathbf{p}}_t + q(1 - \cos(\tilde{\theta}_t))^2 + \tilde{\mathbf{u}}_t^\top R \tilde{\mathbf{u}}_t | \mathbf{e}_\tau = \mathbf{e}\right)\right] \\ \text{s.t.} \quad \mathbf{e}_{t+1} = g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{w}_t), \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\sigma})^2), \quad t = \tau, \tau+1, ... \\ \mathbf{u}_t = \pi(t, \mathbf{e}_t) \in \mathcal{U} \\ \tilde{\mathbf{p}}_t + \mathbf{r}_t \in \mathcal{F} \end{aligned} \quad (3)$$

Here, $\mathbf{Q} \in \mathbb{R}^{2\times 2}$ and $Q \succ 0$ is a symmetric positive-definite matrix defining the stage cost for deviating from the reference position trajectory. The variable $q \in \mathbb{R}, q > 0$, defines the stage cost for deviating from the reference orientation trajectory. The matrix $\mathbf{R} \in \mathbb{R}^{2\times 2}$ and $R \succ 0$ corresponds to the stage cost for using excessive control effort.

## 3 Technical Approach

We approach this infinite-horizon stochastic optimal control problem using a receding-horizon certainty equivalent control (CEC) approach and a generalized policy iteration (GPI) approach. For the CEC approach, we simplify the problem by assuming that noise is fixed at their expected values, and using a non-linear program (NLP) solver to solve a related sub-problem at each timestep. For the GPI approach, we discretize the state and control space so that we can run policy iteration to find a solution.

In both approaches, we account for wrap-around when calculating the angular component of $\mathbf{e}$. For a given angular difference, whenever appropriate we set $\tilde{\theta} \leftarrow (\tilde{\theta} + \pi)\%(2\pi) - \pi$, where $\%$ refers to the modulo operator.

### 3.1 Receding-horizon certainty equivalent control (CEC)

Specifically for the CEC approach, we simplify the problem by assuming that the noise variables $\mathbf{w}_t$ were fixed at their expected values of zero. This turns the stochastic optimal control problem to a deterministic optimal control problem [1] which is more easily solved. Additionally, the receding horizon characteristic approximates the infinite-horizon case. Then at each stage, we find the optimal

control for this simplified problem. We definite the discounted finite-horizon deterministic optimal control problem at each time step as:

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

Here, the function $\mathfrak{q}$ defines a chosen terminal cost. The CEC problem can be algebraically manipulated to fit the NLP of form:

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

where $\mathbf{U} = [\mathbf{u}_\tau^\top, ..., \mathbf{u}_{\tau+T-1}^\top]^\top$ and $\mathbf{E} = [\mathbf{e}_\tau^\top, ..., \mathbf{e}_{\tau+T}^\top]^\top$. We use CasADi [3] to solve our formulated NLP.

Specifically, we setup our NLP as follows. For best results, we chose $\mathbf{Q} = I_2, q = 4, \mathbf{R} = 0.25 I_2$. Additionally, we let $\mathfrak{q}(\mathbf{e}) = ||\mathbf{e}||_2^2$. We chose $\gamma = 0.75$.

The objective function for CasADi is the same objective as given in equation (4). The decision variables for the optimization are $\mathbf{U} \in \mathbb{R}^{2T}$ as defined above. The elements of vector $\mathbf{U}$ alternate as $[v_\tau, \omega_\tau, v_{\tau+1}, \omega_{\tau+1}, ...]^\top$. Correspondingly the lower bound $\mathbf{U}_{lb} \in \mathbb{R}^{2T} = [0, -1, 0, -1, ...]^\top$. And similarly the upper bound $\mathbf{U}_{lb} \in \mathbb{R}^{2T} = [1, 1, 1, 1, ...]^\top$.

The additional constraints are constructed as follows. Continuing from the problem formulation, the robot position can be recovered as $\mathbf{p}_t = \tilde{\mathbf{p}}_t + \mathbf{r}_t$. Then let vector

$$
\mathbf{h} = \begin{bmatrix}
\vdots \\
\mathbf{p}_\tau \\
-\mathbf{p}_\tau \\
(\mathbf{p}_\tau[0] - \mathcal{C}_1[0])^2 + (\mathbf{p}_\tau[1] - \mathcal{C}_1[1])^2 \\
(\mathbf{p}_\tau[0] - \mathcal{C}_2[0])^2 + (\mathbf{p}_\tau[1] - \mathcal{C}_2[1])^2 \\
\vdots
\end{bmatrix}
$$

where the explicitly written block repeats for $t = \tau, ..., \tau+T-1$. The third and fourth expressions in the above block correspond to constraints that the robot do not collide with circular obstacles one and two, respectively, at any time step. Note that this block has size $\mathbb{R}^6$, so the vector $\mathbf{h} \in \mathbb{R}^{6T}$. There is no upper bound constraint $\mathbf{h}_{ub}$, but there is a lower bound constraint:

$$
\mathbf{h}_{lb} = \begin{bmatrix}
\vdots \\
-3 \\
-3 \\
-3 \\
-3 \\
0.5^2 \\
0.5^2 \\
\vdots
\end{bmatrix}
$$

Again, the last two constraints in the block correspond to the avoiding collision with the circular obstacles. The first two constraints, $-3$, correspond to $\mathbf{p}_\tau$ as a lower bound on the free space. The second two constraints, also $-3$, correspond to $-\mathbf{p}_\tau$, which is equivalent to the upper bound on the free space.

With the objective and constraints of equation (5) defined, the NLP is solved using CasADi.

### 3.2 Generalized policy iteration (GPI)

For GPI, we solve equation (3) by discretizing the state and control spaces. We discretize the state space using a position resolution of $0.05$ (in both $x$ and $y$), and an angular resolution of $\frac{2\pi}{100}$. We discretize the control space with a resolution of $0.1$ for both $v$ and $\omega$. Let $(n_t, n_x, n_y, n_\theta)$ and $(n_v, n_\omega)$ be the number of grid points, respectively. As mentioned previously, wrap-around is enforced for angles. Note that the lissajous curve we are following [1] has a period of 100, so $n_t = 100$.

As suggested by [1], we create transition probabilities in the discretized MDP by choosing, for discrete state $\mathbf{e}$ and discrete control $\mathbf{u}$, the next grid points $\mathbf{e}'$ around the mean $g(t, \mathbf{e}, \mathbf{u}, 0)$ and evaluating the likely hood of $\mathcal{N}(\mathbf{0}, diag(\boldsymbol{\sigma})^2)$ at each of the valid grid points. In order to be a valid grid point, the corresponding robot position $\mathbf{p}'_k$ to each $\mathbf{e}'$ must be within the free space $\mathcal{F}$. This includes checking the environment bounds and collision with the circular obstacles.

In our implementation, we check grid points within a 3D block that has half-widths $3\boldsymbol{\sigma}$ of the expected motion $\mathbf{e}'$. The likelihood $\mathcal{N}(\mathbf{0}, diag(\boldsymbol{\sigma})^2)$ is evaluated at each grid point. If the corresponding robot grid positions $\mathbf{p}'_k$ to this $\mathbf{e}'$ is not in $\mathcal{F}$, the evaluated likelihood is set to 0. Then the remaining likelihoods are normalized to sum to 1, so that this becomes a valid transition probability $p_f$.

The generalized policy iteration algorithm is described in Algorithm 5, reproduced from [2] where $\mathcal{B}$ refers to the Bellman backup operator and $H$ refers to the Hamiltonian operator.

**Algorithm 5** Generalized Policy Iteration

1: Initialize $V_0$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \pi_{k+1}(\mathbf{x}) = \underset{\mathbf{u} \in \mathcal{U}(\mathbf{x})}{\arg\min}\, H[\mathbf{x}, \mathbf{u}, V_k(\cdot)]$ ▷ Policy Improvement
4: $\quad V_{k+1} = \mathcal{B}^n_{\pi_{k+1}}[V_k], \quad$ for $n \geq 1$ ▷ Policy Evaluation

Generalized Policy Iteration, reproduced from [2].

Recall from [2] that the Hamiltonian is $H[\mathbf{x}, \mathbf{u}, V(\cdot)] = \ell(\mathbf{x}, \mathbf{u}) + \gamma \mathbb{E}_{\mathbf{x}' \sim p_f(\cdot|\mathbf{x},\mathbf{u})}[V(\mathbf{x}')]$. In our case, we instead use the error state $\mathbf{e}$ in place of $\mathbf{x}$. And, our stage cost is $\ell(\mathbf{e}, \mathbf{u}) = ||g(t, \mathbf{e}_t, \mathbf{u}_t, \mathbf{0})||_2$ for appropriate $t$. The $p_f$ for the expectation here is the transition probability described above.

We implement an online version of policy iteration. So $H[\mathbf{e}, \mathbf{u}, V(\cdot)]$ is only evaluated as necessary during run time, for values of $\tau, \mathbf{e}, \mathbf{u}$ that we encounter. This reduces computation greatly because we do not need to explore states that are far from the reference trajectory.

After running policy iteration until it converges, the optimal control policy $\pi(\tau, \mathbf{e})$ is returned. This can be used at each timestep $\tau$ and error state $\mathbf{e}$ of the simulation to find the optimal control inputs.

## 4 Results

### 4.1 Receding-horizon certainty equivalent control (CEC)

Good results were obtained using the CEC approach. Results are shown in Figure 2. Additionally, a GIF file of the trajectory following is included with the code submission [1].

While the reference trajectory goes near the top right obstacle $\mathcal{C}_2$ and goes through the bottom left obstacle $\mathcal{C}_1$, the robot manages to avoid both obstacles during its tracking of the reference trajectory. Looking at the animation of the robot, it tends to fall behind the reference trajectory whenever the reference trajectory has a sharp turn near (or inside) an obstacle. Namely, when approaching the top right and bottom left parts of the target curve, the robot "pauses" for a few steps while it turns in place to avoid the obstacle. Then, in order to catch up to the reference trajectory, the robot moves forward with high linear velocity. In any case, the overall robot's trajectory visually seems very close to the target trajectory while also avoiding collision with the obstacles. There does not appear to be any significant effects due to ignoring the noise in the CEC formulation.

This CEC approach completed with the following results:

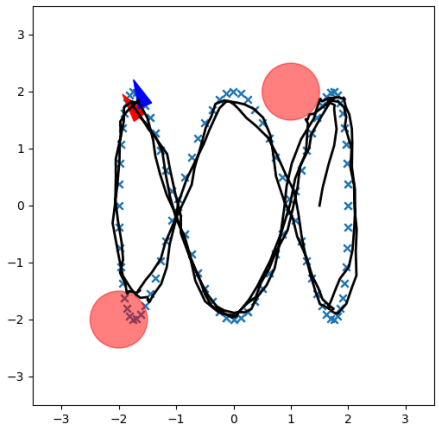


Figure 1: The robot (red triangle) tracks a reference trajectory (blue triangle, cyan crosses) and avoids the circular obstacles (red). The robot's trajectory is shown as the black line. The robot begins at position $(1.5, 0.0)$, displaced from the reference trajectory. This trajectory corresponds to $T = 10$.

| | $T = 5$ | $T = 10$ | $T = 15$ |
|---|---|---|---|
| Total time | 5.680 s | 16.330 s | 33.550 s |
| Average iteration time | 23.605 ms | 67.986 ms | 139.737 ms |
| Final error | 86.276 | **85.269** | 88.930 |

Table 1: CEC approach performance.

Overall, the final errors for $T = \{5, 10, 15\}$ are similar. The NLP solver ran quickly and produced the best results when $T = 10$. Lower $T$ values have faster computation, while higher $T$ values have slower computation. Here, $T = 10$ produced the best results. This is likely because we chose a relatively high value of $\gamma = 0.75$. So with larger $T$, values of future locations of the reference trajectory are considered more than if $\gamma$ was small. This may cause the robot to over-compensate for future moves and cause a high current error in its trajectory.

Looking at Figure 2, one notable aspect of the robot's trajectory is that it sometimes turns relatively sharply. This is because from one timestep to the next, the robot has input to it's angular velocity anywhere from $[-1, 1]$. In future work, it may be interesting to see if the robot's overall trajectory (including turns) can be made smoother. One possible solution would be to take smaller timesteps for the robot than the default 0.5 [1], and at the same time reduce the angular velocity input space to be closer to 0.

### 4.2 Generalized policy iteration (GPI)

There is some error in my implementation of GPI, which results in a failure to track the reference trajectory. The robot seems to wiggle in place without following the desired trajectory. In some timesteps, it appears to move backward a little due to noise in the motion model. The calculation of the transition probabilities appears to be correct. The function for that calculation was tested using values for the error and reference states on the border of $\mathcal{C}_2$. The transition probabilities are 0 corresponding to when the robot position would be inside the circle.

Because this policy iteration was implemented in an online manner to reduce memory consumption, it is likely that I am updating the value function $V$ or the policy $\pi$ incorrectly. In its current state of implementation, this algorithm takes around 10 minutes to run. Since it is slower than the CEC ap-

proach, and more difficult to implement correctly, the CEC approach to the originally state problem in equation (3) seems to be superior.

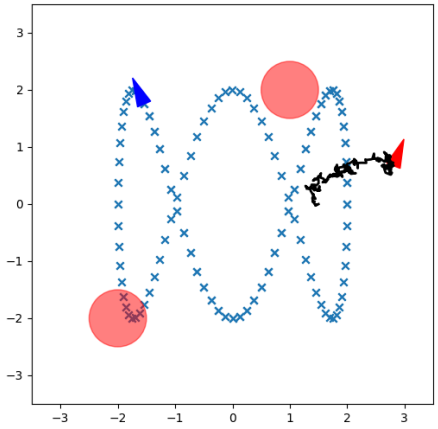


Figure 2: The robot (red triangle) fails to follow a reference trajectory (blue triangle, cyan crosses). The robot begins at position $(1.5, 0.0)$, displaced from the reference trajectory.

If there were more time for further research, I would continue to debug the current implementation or also try additional solutions to this problem. Other solutions include ideas such as adaptive discretization of $\mathbf{e}$ and feature-based representations of the value function [1] would be considered.

## References

[1] Nikolay Atanasov. Project 3: Infinite-horizon stochastic optimal control. In *ECE276B: Planning & Learning in Robotics*, 2021.

[2] Nikolay Atanasov. Lecture 11: Bellman equations. In *ECE276B: Planning & Learning in Robotics*, 2021.

[3] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.