# [Re] Online Optimal Control with Linear Dynamics and Predictions: Algorithms and Regret Analysis NeurIPS Reproducibility Challenge 2019

**Feras Al Taha**
Department of Electrical
and Computer Engineering
McGill University
feras.altaha@mail.mcgill.ca

**Yu Ting Gu**
Department of Computer Science
McGill University
yu.t.gu@mail.mcgill.ca

**Rebecca Xiong**
Department of Electrical
and Computer Engineering
McGill University
yubei.xiong@mail.mcgill.ca

## 1 Introduction

Receding horizon control (RHC) is a control methodology used in several process industries as it handles constraints in optimality problems better than classical control methods [1]. At the cost of a degree of suboptimality, its success in industry has been widely acknowledged over the past years and its popularity growth is not slowing. The strategy consists in computing at every step the optimal set of actions to take over a fixed time horizon [2], with the purpose of controlling the system over a larger (possibly infinite) horizon. The first obtained action is taken and at the next time step, a new set of actions is computed again. Accordingly, at every moment, the RHC method solves an optimization problem over a finite moving time window. The method is also called model predictive control (MPC) because it takes into account estimates of future quantities based on the information that is currently available, effectively *predicting* the near horizon.

This online optimization problem (where present decisions are made with incomplete knowledge of the future) is considered with time varying costs by Li *et al.* [3]. Using a gradient-based algorithm, they solve for the control policy to use on the given system and evaluate it using dynamic regret, or the difference between the algorithm's performance and the actual optimal performance. This report attempts at reproducing the numerical results presented for a linear quadratic (LQ) tracking problem and a non-linear tracking example [3] as well as exploring the effects of modifications on the algorithm and the examples used via some experiments.

## 2 Problem formulation

Consider a linear time invariant (LTI) dynamical system with state $x_t$, input $u_t$ and system matrices $(A, B)$ for which a performance function is defined with time-varying cost $f_t(x_t) + g_t(u_t)$ and terminal cost $f_N(x_N)$. Then, the corresponding optimal control problem can be formulated as a minimization problem with constraints defined by the system dynamics:

$$\min_{x,u} J(x,u) = \min_{x,u} \sum_{t=0}^{N-1} [f_t(x_t) + g_t(u_t)] + f_N(x_N)$$
$$\text{s.t.} \quad x_{t+1} = Ax_t + Bu_t, \quad t \geq 0$$

Preprint. Under review.

That being said, the problem can be reformulated as an unconstrained optimization problem, by rewriting the state as $z_t := (x_t^{k_1}, \ldots, x_t^{k_m}),\ t > 0$ where $k_j$ are the row indices of non-zero rows in the $A$ matrix which is assumed to be in canonical form [3]. Accordingly, the control input can be solved for with the expression $u_t = z_{t+1} - A(\mathcal{I}, :)x_t$ where $A(\mathcal{I}, :)$ consists of the non-zero rows of $A$ (i.e. rows $k_1, \ldots, k_m$). With the new reformulated optimization problem, an objective function can be defined with the original time varying cost functions, the controllability index $p$ and the new variable $z$:

$$C(z) := \sum_{t=0}^{N} f_t(z_{t-p+1}, \ldots, z_t) + \sum_{t=0}^{N-1} g_t(z_{t-p+1}, \ldots, z_{t+1}) := \sum_{t=0}^{N} f_t(x_t) + \sum_{t=0}^{N-1} g_t(u_t)$$

Rather than optimizing over the full interval from $t = 0$ to $t = N$, RHC will consider optimizing over a smaller window or horizon $W$ as it will be more computationally efficient and will allow to solve for a suboptimal solution despite the lack of knowledge of the future a priori. Accordingly, using the gradient of the partial cost over a finite lookahead window will allow to design an online algorithm for gradient descent.

$$\frac{\partial C(z)}{\partial z_t} = \sum_{\tau=t}^{t+p-1} \frac{\partial f_\tau}{\partial z_\tau}(z_{\tau-p+1}, \ldots, z_\tau) + \sum_{\tau=t-1}^{t+p-1} \frac{\partial g_\tau}{\partial z_\tau}(z_{\tau-p+1}, \ldots, z_{\tau+1})$$

To minimize the defined cost $C(z)$, the gradient descent is constructed with four hyper-parameters $\delta_c$, $\delta_y$, $\delta_\omega$, and $\delta_z$ [4]. For each $j^{th}$ iteration ($j \geq 0$), the gradient descent updates the states as follows:

$$\begin{aligned}
\boldsymbol{\omega}(j+1) &= (1+\delta_\omega)\boldsymbol{\omega}(j) - \delta_\omega \boldsymbol{\omega}(j-1) - \delta_c \nabla C(\boldsymbol{y}(j)) \\
\boldsymbol{y}(j+1) &= (1+\delta_y)\boldsymbol{\omega}(j+1) - \delta_y \boldsymbol{\omega}(j) \\
\boldsymbol{z}(j+1) &= (1+\delta_z)\boldsymbol{\omega}(j+1) - \delta_z \boldsymbol{\omega}(j)
\end{aligned}$$

where $\boldsymbol{\omega}(j), \boldsymbol{y}(j)$ are auxiliary variables to accelerate the convergence, and $\boldsymbol{z}(j)$ is the output variable. By choosing which parameters to set as zero, different gradient descent methods can be chosen, as described in Table 1.

Table 1: Stepsizes for each gradient descent method [4]

| **Method** | **Parameters** ($\delta_c$, $\delta_y$, $\delta_\omega$, $\delta_z$) |
|---|---|
| Gradient Descent | ($\delta_c$, 0, 0, 0) |
| Heavy Ball Method | ($\delta_c$, $\delta_y$, 0, 0) |
| Nesterov's Accelerated Gradient | ($\delta_c$, $\delta_y$, $\delta_y$, 0) |
| Triple Momentum Method | ($\delta_c$, $\delta_y$, $\delta_\omega$, $\delta_z$) |

At every time step $t$, the following optimization is executed to obtain the control to apply and progress the system ahead, defining the Receding Horizon Gradient-based Control (RHGC) algorithm:

1. Initialize $z_{t+W}$ where $W$ is the lookahead window or horizon.
2. Perform one iteration of the gradient descent method (with vanilla gradient descent, Nesterov's accelerated gradient or triple momentum) to update $z_{t+W}$ until $z_{t+1}$.
3. Using $z_{t+1}$ and the observed state $x_t$, compute the control input $u_t = z_{t+1} - A(\mathcal{I}, :)x_t$ to be applied.

To measure the performance of the online algorithm, the regret metric (also known as competitive difference or dynamic regret) can be used. It is defined to be the difference between the performance function $J(x, u)$ evaluated at the inputs computed by the suggested algorithm and its value when evaluated at the optimal solution.

$$Regret(x, u) := J(x, u) - J^*$$

where $x, u$ are the states and inputs obtained by the algorithm and $J^*$ is the optimal performance function.

The LQ tracking problem considers an LTI system where the cost is defined such any deviation of the state from a reference trajectory results in a penalty while still taking into account the input.

$$J(x,u) = \frac{1}{2}\sum_{t=0}^{N-1}[(x_t-\theta_t)^T Q_t(x_t-\theta_t) + u_t^T R_t u_t] + \frac{1}{2}(x_N-\theta_N)^T Q_N(x_N-\theta_N)$$

where $\theta_t$ for $t \in [0, N]$ is the reference trajectory to follow and $Q_t$, $R_t$, $Q_N$ are the state trajectory deviation cost, the input cost and the terminal state cost respectively. When the reference trajectory is not fully known a priori and is instead revealed progressively, the problem becomes an online control problem.

For the non-linear example, a path tracking problem involves a two-wheel mobile robot with nonlinear kinematic dynamics $\dot{x} = v\cos\delta$, $\dot{y} = v\sin\delta$, and $\dot{\delta} = \omega$, where $(x, y)$ are the coordinates of the robot, $v$ and $w$ are the tangential and angular velocities, and $\delta$ denotes the tangent angle between $v$ and the $x$ axis. Given a reference path $(x_t^r, y_t^r)$, by applying control on $v$ and $w$, the objective is to minimize the control cost:

$$Cost = \sum_{t=0}^{N}[c_t \cdot (\Delta x_t^2 + \Delta y_t^2) + c_t^v \cdot v_t^2 + c_t^\omega \cdot \omega_t^2],$$

where $\Delta x_t = x_t - x_t^r$, and $\Delta y_t = y_t - y_t^r$. $c_t$, $c_t^v$, and $c_t^\omega$ are the cost coefficients defined in Appendix H of the paper.

## 3 Experimental Methodology

Li *et al.* present and consider three RHGC methods: standard Gradient Descent (RHGD), Gradient Descent with Nesterov's Accelerated Gradient (RHAG), and Gradient Descent with Triple Momentum (RHTM). They compare these against a sub-optimal model-predictive control algorithm (subMPC) which combines online computation of the optimal solution with offline derivation of an explicit control law using a piecewise affine approximation [5] with varying number of iterations.

The authors show in Section 4 of their paper that RHGD and RHTM will improve the performance exponentially as the horizon gets larger regardless of the initialization oracle used [3]. We try to reproduce this exponential decay of the regret and compare it against the same subMPC instances used. For the numerical examples executed by the authors, they define and use Follow the Optimal Steady State (FOSS) as an initialization oracle, which has a provable bound on the regret with RHTM, so we use the same oracle. We also note that the Regret of the initialization oracles are dependent on the cost functions, which are time-variant and randomly generated in the code. Given that experimenting with different initializers did not give us much different results, these results will not be reported. Finally, we try another gradient descent method not mentioned by the authors, which is the Polyak method, also known as the Heavy Ball method [4] (for which the corresponding RHGC algorithm will be referred to as Receding Horizon Heavy Ball (RHHB)), using $\delta_c = 4/(\sqrt{L}+\sqrt{m})^2$, $\delta_y = (\sqrt{L}-\sqrt{m})/(\sqrt{L}+\sqrt{m})$, and $\delta_\omega = \delta_z = 0$ as input to the provided code.

While the numerical examples in the paper involve stable systems only (i.e. for which the output response is bounded), the stability of the system is not a required assumption of the method. Thus, unstable system matrices $A$ should in theory be handled as well. Therefore, we perform the same LQ problem experiment but using an unstable system.

In addition to the LTI system for which the algorithm is applied, the paper shows that the RHGC methods are capable of handling nonlinear systems. In particular, experiments were done for path tracking of a two-wheel mobile robot, where nonlinear kinematic dynamics are considered. With a heart-shaped reference trajectory, discretized by step size $\Delta t = 0.025s$, it is shown that the robot follows the reference trajectory well especially on smoother part of the heart but worse on its sharp corners, and a longer horizon ($W$) leads to better tracking performance. It is an arbitrary choice to use $\Delta t = 0.025s$, and only results for two different horizons ($W = 1s$ and $W = 2s$) are shown. Hence, we replicate the same experiment with varying combinations of discrete-time resolution $\Delta t$ and $W$, and extend the test case with a continuously differentiable petal-shaped trajectory with no sharp corners. As opposed to a pure visualisation of the path-tracking, we also provide a more quantitative analysis on the performance of the algorithm using metrics such as the overall cost and Root Mean Square Error (RMSE).

# 4 Results

## 4.1 LQ Tracking

For the LQ tracking problem, we see that the performance of subMPC improves with an increased number of iterations, but saturates eventually whereas the RHGC methods continue to improve with exponentially decaying regret as the horizon window, $W$, is increased. This is all consistent with the original paper. Figures 1 and 2 showcase the performance of each method for different horizons. This result is consistent for stable $A$. However, when we look at unstable systems, we see that RHHB

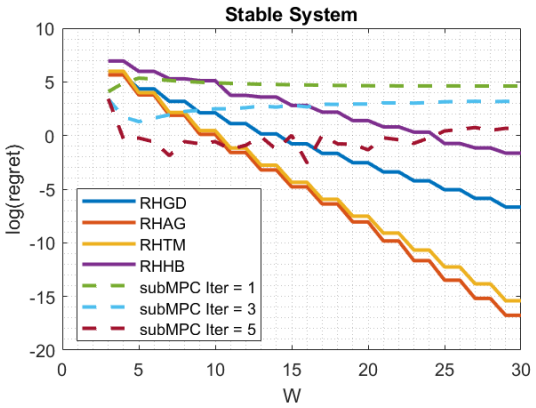



Figure 1: Regret on a stable system.

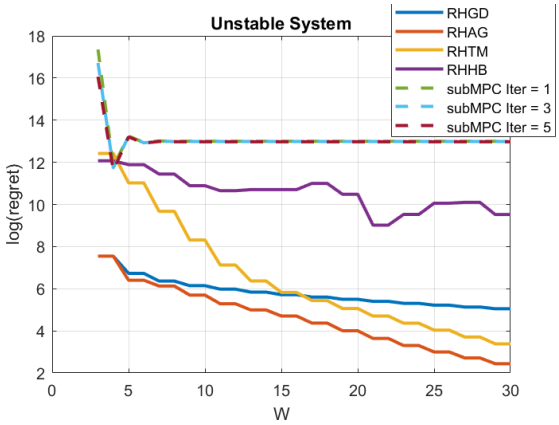



Figure 2: Regret on an unstable system.

performs inconsistently, and RHTM's performance is not as good relative to RHAG and RHGD for small horizons. The bad performance on RHHB is attributed to the weakness of the heavy ball method itself, which is the possibility of converging to a local minimum, but not the global minimum during its gradient update [4]. Thus, we shift our focus to comparing the performance of the other three RHGC methods, considering results for the stable systems and unstable systems separately.

### 4.1.1 Stable System

Using the code provided by the authors and the same stable matrix ($A = [0, 1; -\frac{1}{6}, \frac{5}{6}], B = [0; 1]$) and step sizes mentioned in Appendix H, we are able to reproduce the same figure as in their paper. However, given that the cost matrices $Q_t$ and $R_t$ are randomly generated and thus define a different optimization problem setup at every execution, RHTM will sometimes perform slightly worse than RHAG due to the fluctuations caused by the extra momentum present in the triple momentum, as illustrated in Figure 3. That said, the observed decay is still exponential with respect to the look-ahead window which is in line with the claim presented in the paper.

### 4.1.2 Unstable System

We run the same tests but on an unstable system defined by $A = [0, 1; 6, -4], B = [0; 1]$, where we observe that RHTM performs worse relative to RHAG. Compared to RHGD, we see that the RHTM's regret is worse than RHGD at smaller horizons, until we reach a horizon of 16. This matches the theoretical regret upper bounds described in the paper.

The decay of regret is still exponential, but the initial regret is very high in comparison to the other to RHGC methods as illustrated in Figure 4. According to Theorem 1 in their paper, the bound of regret for RHTM and RHGD for an initialization oracle $\phi$ is expressed as

$$Regret(RHGD) \leq \zeta \left(\frac{\zeta - 1}{\zeta}\right)^K Regret(\phi), Regret(RHTM) \leq \zeta^2 \left(\frac{\sqrt{\zeta} - 1}{\sqrt{\zeta}}\right)^{2K} Regret(\phi)$$

where $K = \left\lfloor \frac{W-1}{p} \right\rfloor$, $Regret(\phi)$ is the regret of the initializer $\phi$, $p$ is the controllability index of the system (matrices $A$ and $B$), and $\zeta$ is the condition number of the cost matrix. The authors do not

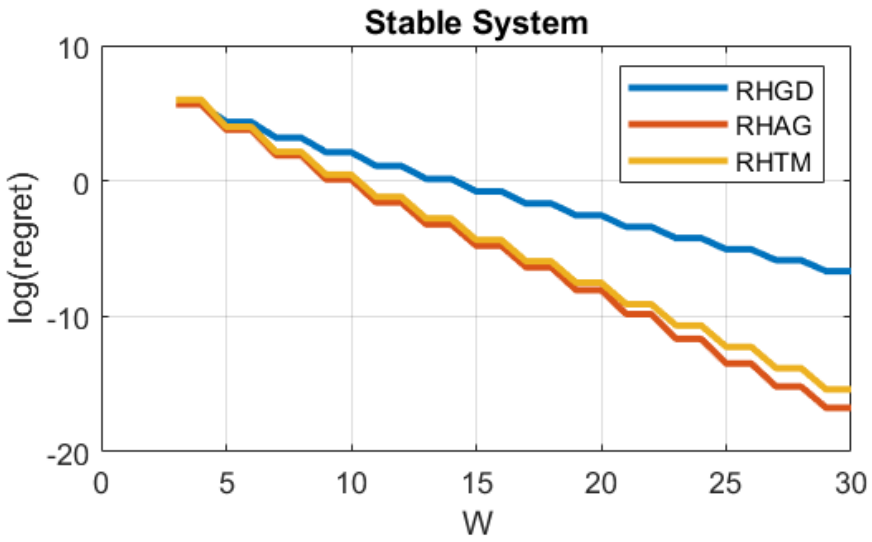



Figure 3: Regret at different horizons for a stable system. Note that in this particular instance, RHAG performs better and has faster regret decay than RHTM.

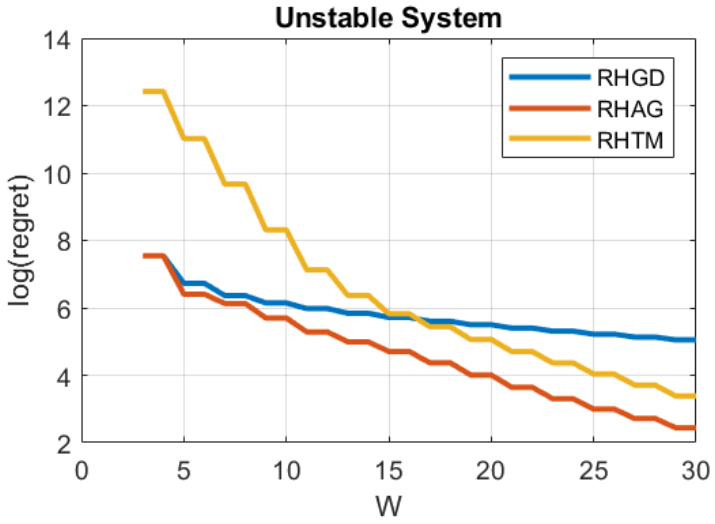



Figure 4: Regret at different horizons for an unstable system. We see RHTM starts with a much larger regret, but converges back to a similar regret to two other methods with a large enough look-ahead.

provide a bound for $Regret(RHAG)$ in the paper, so we are unable to comment on it. In our example, $\zeta = 46$, $p = 2$, and $W$ is the horizon window we are plotting against. Regret($\phi$) is the same on both sides since the same initializer is used, so we focus on the coefficients. We notice that RHTM follows a stronger exponential decay compared to RHGD, but has a much higher regret at low $W$. This makes sense, since $\zeta$ is very high, causing the high regret at low $W$. Since $K$ increases linearly with $W$, the bound on regret will decrease exponentially with respect to $W$, and we see this allows TM to overtake RHGD when $W$ is sufficiently large. Finally, we observed that in an unstable system, $\zeta$ seems to always be much larger than for a stable system, so we expect this result to be consistent regardless of the choice of unstable $A$.

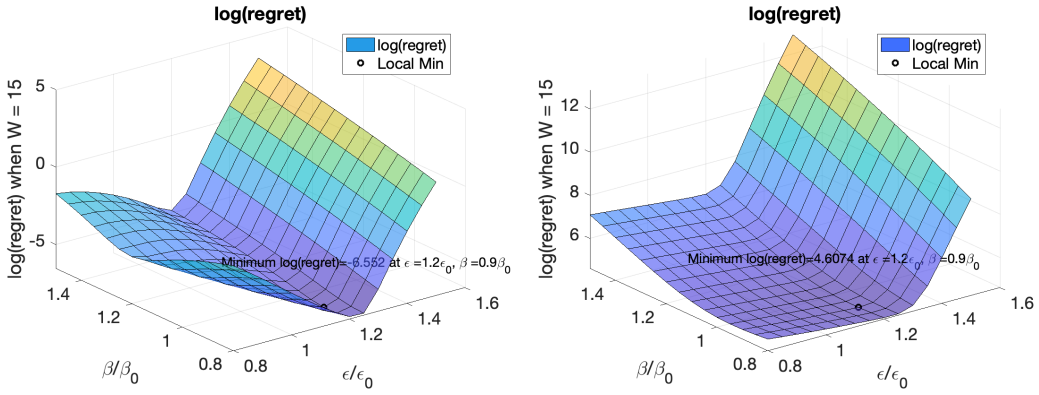
Finally, a grid search is performed to verify the effects of hyper-parameters in the gradient descent and locate the best set of step-sizes to minimize the regret. Since the performance of RHTM fluctuates and involves all four step-sizes, we only perform the test for RHAG with $\delta_c = 1/l_c$, $\delta_y = \delta_\omega = \frac{\sqrt{\zeta}-1}{\sqrt{\zeta}+1}$ and $\delta_z = 0$. We denote the two hyperparameters default values of $\delta_c$ and $\delta_y = \delta_\omega$ as $\epsilon_0$ and $\beta_0$

respectively, and tested the effects of their variation on the regret function by scaling both by a factor varying from 0.8 to 1.4:

$$\text{i.e. } \delta_c = \epsilon = k_c \epsilon_0 \text{ and } \delta_y = \delta_\omega = \beta = k_y \beta_0$$

$$\text{where } \epsilon_0 = \frac{1}{l_c}, \beta_0 = \frac{\sqrt{\zeta} - 1}{\sqrt{\zeta} + 1}, \text{ and } k_c, k_y \in \{0.8, 0.85, \ldots, 1.35, 1.4\}$$

Figures 5a and 5b show the varying $log(Regret)$ when $W = 15$ with different step-size combinations for the stable system and the unstable system, respectively. We use the same system matrices $(A, B)$ as in the experiments shown in Figure 3 and 4. It is noted that the regret is minimized when $\epsilon = 1.2\epsilon_0$ and $\beta = 0.9\beta_0$ in both stable and unstable system, implying the gradient descent stepsizes used by the authors are not far from the optimal settings but could be adjusted to further minimize cost. Due to the randomness of the cost matrices $Q$ and $R$, the results may vary a little.




(a) Stable System (b) Unstable System

Figure 5: $log(Regret)$ when $W = 15$ with varying step-sizes $\epsilon$ and $\beta$, for RHAG

### 4.2 Two-wheel robot tracking with nonlinear dynamics

Based on the source code provided by the authors, we modified the control time resolution of the reference trajectory. In addition to $\Delta t = 0.025s$, which is the step size used in the paper, we tested with $\Delta t = 0.1s$ and $\Delta t = 0.05s$, and changed the horizon $W$ in all three cases to verify whether increasing $W$ could truly make the model follow the trajectory better. The performance of the model is evaluated based on two criteria: 1) the objective cost function defined in the paper; 2) the RMSE which measures how far the prediction deviates from the reference path

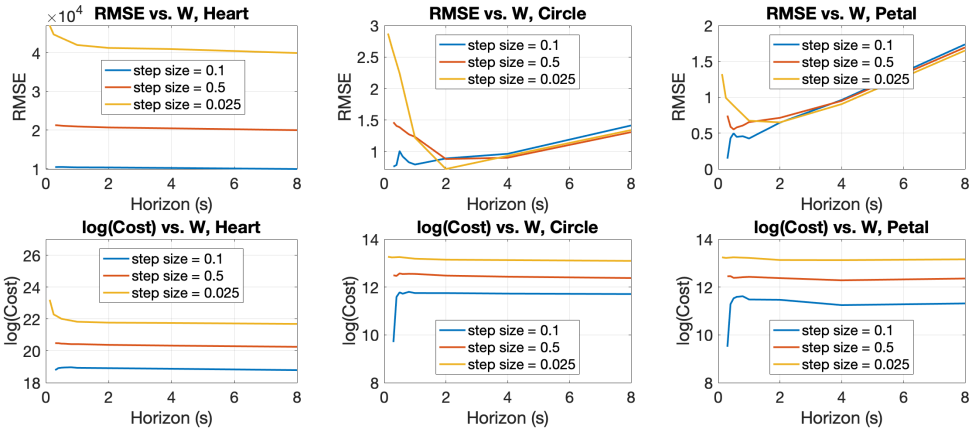
$$RMSE = \frac{\sum_{t=0}^{N} \sqrt{\Delta x_t^2 + \Delta y_t^2}}{N}$$

Tests are run for three different reference paths, including the *Heart* and the *Circle*, which are provided in the original source code, as well as an extra *Petal* trajectory which we added. This additional *Petal* trajectory $(x_t^r, y_t^r)$ is defined as

$$r = 20 \sin 0.2t \qquad x_t^r = r \cos 0.1t \qquad y_t^r = r \sin 0.1t$$

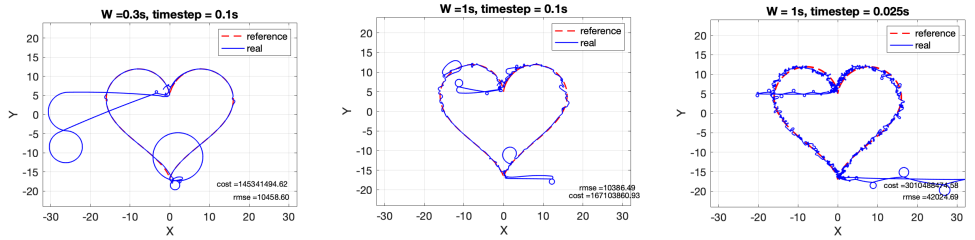
The results are shown in Figure 6a, 6b, and 6c for the three paths respectively. In the *Heart* example, we could observe that when $\Delta t = 0.025s$, bigger lookahead window leads to lower RMSE, thus better overall performance, as noted by the authors. This observation however does not hold for the other two reference paths, where there exists a global minimum of the RMSE with respect to $W$, and increasing $W$ beyond the global minimum generally leads to poorer performance. To explain this, we further examine the behavior of the robot when $W$ is varied in difference cases.

Figure 7a, 7b, 8a and 8b illustrate the robot trajectory relative to the reference paths with $W$ increased from $0.3s$ to $1s$. It is noted that a smaller $W$ allows the robot to follow the reference path more closely, introducing less error due to less frequent oscillations. However, corners in the trajectory are



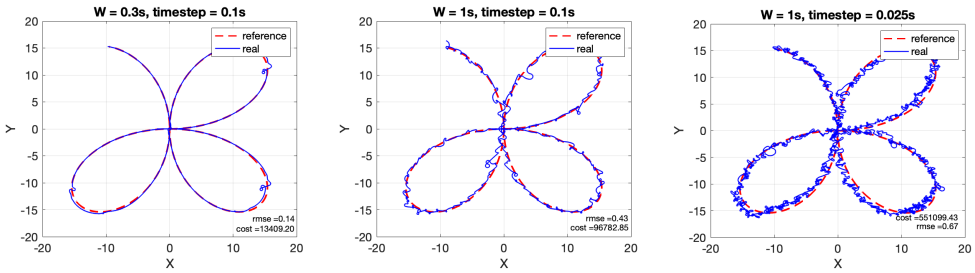

(a) Heart (b) Circle (c) Petal

Figure 6: RMSE and Cost vs. Horizon




(a) $W = 0.3s, \Delta t = 0.1s$ (b) $W = 1s, \Delta t = 0.1s$ (c) $W = 1s, \Delta t = 0.025s$

Figure 7: Heart Trajectory




(a) $W = 0.3s, \Delta t = 0.1s$ (b) $W = 1s, \Delta t = 0.1s$ (c) $W = 1s, \Delta t = 0.025s$

Figure 8: Petal Trajectory

handled less well, where the robot diverges away from the reference, creating large errors. A larger $W$ makes the robot oscillate around more frequently while following the reference path but handles the corner cases more robustly, as illustrated on the sharp corners of the *Heart*. In the *Petal* case, since the reference path is always smooth, lower $W$ is actually desirable to keep the robot following closely to its reference. Therefore, as opposed to the conclusion stated in the paper, larger $W$ improves the tracking performance and is preferable only when the reference path involves singular points.

Moreover, for smaller $\Delta t$, more oscillations were present in the robot's trajectory, as illustrated in Figures 7c and 8c. An explanation for this is that with different control time resolutions, the gradient step size needs to be tuned accordingly, as it depends on the Lipschitz smoothness factor of the cost function which itself is affected by the control time step. As the gradient step-sizes we used

were fixed and not tuned, it may not be the proper choice for different control time resolutions. In addition, it is also possible that the non-convexity of the cost function causes numerical issues when the parameters are not chosen properly.

### 4.3 Conclusion

We are able to successfully reproduce Figure 1 and 2 from [3] with the provided code. For the LQ tracking problem, we verify that the RHGC methods proposed perform much better and converge much quicker than the subMPC methods, for both a stable and unstable system, though we notice fluctuations in performance of the TM method that was not mentioned in the paper. We also try the heavy-ball method which does not yield better results. Additionally, we use grid search to tune the hyper-parameters used in the AG method and find that for both stable and unstable system, the regret can be further reduced when the stepsizes $\delta_c$ and $\delta_y$ are changed to 1.2x and 0.9x of their proposed values in the paper. For the robot path-tracking problem, using the provided code, we reproduce the experiments on the *Heart* trajectory and observe better results with larger horizon, matching the conclusions made by the authors. However, for smooth trajectory, such as the *Circle* and the *Petal*, there exists a relatively small optimal $W$ where the RMSE can be minimized. In addition, although we expected that higher control time resolution would improve performance, we find that increasing the resolution introduces more oscillations, and worsens tracking performance.

### Acknowledgments

We would like to thank Yingying Li for answering our questions and offering insight and advice about the results obtained with our experiments as well as clarifying some theoretical aspects of the paper.

## References

[1] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice — a survey," *Automatica*, vol. 25, pp. 335–348, May 1989.

[2] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52–65, 2011.

[3] Y. Li, X. Chen, and N. Li, "Online optimal control with linear dynamics and predictions: Algorithms and regret analysis," in *Advances in Neural Information Processing Systems*, pp. 14858–14870, 2019.

[4] B. Van Scoy, R. A. Freeman, and K. M. Lynch, "The fastest known globally convergent first-order method for minimizing strongly convex functions," *IEEE Control Systems Letters*, vol. 2, pp. 49–54, Jan 2018.

[5] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization," *IEEE Transactions on Automatic Control*, vol. 56, pp. 1524–1534, July 2011.