An Investigation of Dual MPC with Reinforcement Learning

Aayush Sharma
Aerospace Engineering Department
Georgia Institute of Technology
Atlanta, GA
aayush.sharma374@gatech.edu

Steven Rivadeneira
Aerospace Engineering Department
Georgia Institute of Technology
Atlanta, GA
stevenrr@gatech.edu

Jose Valderrama
Mechanical Engineering Department
Georgia Institute of Technology
Atlanta, GA
jvalderr@gatech.edu

***Abstract*—In this paper, a dual model predictive control algorithm is introduced and investigated. The method is coined "Dual Model Predictive Control with Reinforcement Learning" or "RLdMPC", [1]. This method applies approximate dynamic programming (ADP) to dual model predictive control algorithms to define improved control policies for an agent. This agent will follow suboptimal control policies generated through the estimation of a hyper-state over a finite horizon.**
The RLdMPC method is evaluated against dynamical and electrical models, varying horizon lengths, and measurement estimates. We see how the results of this system show the potential to implement RdMPC algorithms in less-than-ideal and real world systems to provide near-optimal solutions to the cost function minimization implemented.

***Index Terms*—Dual Control, State Estimation, Reinforcement Learning, Adaptive Control, Model Predictive Control**

I. Introduction

A. Approximate Dynamic Programming

The primary goal of Reinforcement Learning (RL) is to dictate the future actions of an agent through an algorithm to determine the necessary actions in the present needed to meet a desired goal. Supervised learning can be achieved by comparing a supervisory signal to your training data. Unsupervised learning strives to understand the structure of data and inferring the characteristics of underlying probability distributions. Reinforcement learning adds rewards to optimal actions by implementing a actor-critic dynamic. This is where the Dual MPC with RL (RLdMPC) algorithm derives its motivation. The dynamic programming approach takes into account a discrete time system with the following attributes

$$x_{k+1} = f_k(x_k, u_k, w_k), k = 0, 1, ...N-1$$

where x_k is the state containing the relevant information the system given a problem, u_k is the action chosen at time k given a set of possible actions (U_k), w_k is a parameter that represents noise in the system, and N is the horizon or number of times the control is applied in the system. At each time step, the system evaluates a cost function given by

$$E\left(c_N(x_N) + \sum_{k=1}^{N-1} c_k(x_k, u_k, w_k)\right)$$

This system will look to minimize the cost function shown above in order to reach the next state. This is done by performing a series of actions governed by a policy, π.

$$\pi = \mu_0, ..., \mu_{N-1}$$

where $u_k = \mu_k(x_k) \in U_k$. This policy is evaluated by the expected cost function

$$J_k^{\pi}(x_0) =_{u \in U(x)} E\left(c_N(x_N) + \sum_{k=1}^{N-1} c_k(x_k, \mu_k(x_k), w_k)\right)$$

The challenge behind this approach is finding a good approximation $J_k^{\pi}(x_0)$ when presented with uncertainty.

The final path will be one that minimizes J for all policies. As the complexity of the problem grows, the feasibility of storing the optimal value function decreases. At this point, the approximate dynamic programming approach is used.

B. Dual Control and its Extensions

The first instance of the dual control problem, and the stochastic adaptive optimal dual control problem was first posed by A. A. Fel'dbaum [7]. These papers gave the foundation to the dual control problem, and Bjorn Wittenmark gives an overview of the different techniques of formulating and solving the adaptive dual control problem optimally and suboptimally [4]. Using the dual control problem posed by Fel'dbaum, and the implementation of Adaptive Dual Control by Wittenmark [5], a Dual Model Predictive Controller algorithm is formulated by Heirung et. al [8]. To give an idea on the scheme between a traditional adaptive control system, and the dual adaptive control system, see Fig. 1 and Fig. 2 [6].

This adaptive dual control system is then applied to the finite horizon optimal control problem, to give the basis of the Dual Model Predictive Controller (DMPC). Dual Model Predictive Control optimally combines plant excitation and control based on current and predicted parameter estimation errors. The DMPC algorithm differs from a standard Model Predictive Controller in that the system state is given by the hyperstate. The combining of Model Predictive Control and State Estimation to give DMPC is outlined in the Problem formulation.

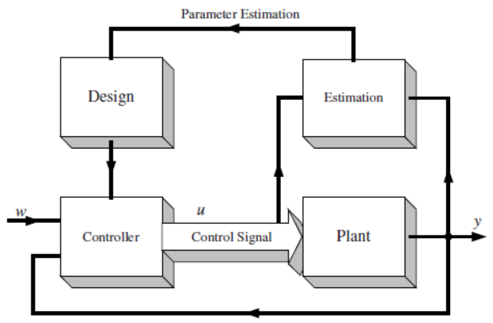

Fig. 1. Adaptive Control System

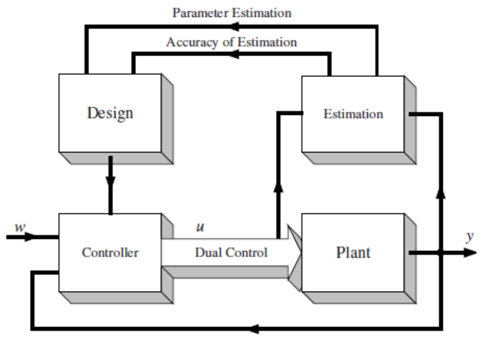

Fig. 2. Adaptive Dual Control System

C. Reinforcement Learning

Reinforcement Learning (RL) has been used widely throughout research to simulate a learning environment for an agent. This agent can arrive to analytical solutions given some uncertainty by maximizing a reward function, $R(s)$. Reward functions characterize the actions, environment and namely the effects each action has within the environment. By modeling the environment as a Markov Decision Process (MDP), the agent may maximize this reward function in order to choose the next best step; the process of maximizing (or minimizing) these given functions is the learning process. On many occasions, deriving a model for the dynamic environment with which an agent may interact can quickly become too computationally expensive. Given a reward function and sufficient time to explore the environment, machines can derive the policy which maximizes the reward function and produces the desired results without having to derive a model of the environment analytically.

In the context of controls, there have been several recent applications in robotics: the famous autonomous helicopter developed by Andrew Ng maneuvered by RL based controls [10], a RL based guidance and control algorithm for six degree of freedom planetary landing [12], and missile homing phase interception using only target line-of-sight angle measurements and RL [12]. Reinforcement learning has proven to be a beneficial tool for control of complex systems. Still, more traditional control methods are often preferred for aircraft controls in practice. Reinforcement learning relies on optimal policies generated from an agent's interaction with its environment. It falters when interactions with the environment are limited. This may be why RL efforts have been mainly focused on robotics. There are many examples, robot manipulators [15], HVAC systems [13], and underwater cable tracking robots [14], all of which use RL based control.

These systems leverage one of two methods: model-based RL and model-free RL. Model-based RL methods evaluate a reward function to perform the most desirable future actions, $\pi(s)$. Model-free RL puts less weight on the uncertainty of the environment. The RL method to be explored in this paper will mimic Q-learning, a model-free RL approach originally developed by Watkins in his paper in 1989. [9]

II. Problem Formulation

A. Unicycle Model

A unicycle model is first used to evaluate the RLdMPC algorithm. The unicycle model has state vector φ, composed of 3 states: horizontal position (x), vertical position (h), and the heading angle (ψ). The unicycle may be controlled with input vector u, composed of linear velocity (v) and angular velocity (ω). It is assumed that the unicycle has access to sensor measurements of horizontal position, vertical position, and heading. To differentiate the internal state φ = (x, h, ψ) from the sensor measurement y, a subscript "m" is added to the measurement states. Lastly, it is assumed that the sensor has some sensor noise (ν). The unicycle state space equations can be summarized as follows:

$$\varphi = \begin{bmatrix} x \\ h \\ \psi \end{bmatrix}, y = \begin{bmatrix} x_m \\ h_m \\ \psi m \end{bmatrix}, u = \begin{bmatrix} v \\ \omega \end{bmatrix}, \theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\dot{\varphi} = \begin{bmatrix} \dot{x} \\ \dot{h} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \end{bmatrix} \tag{2}$$

$$y_t = \theta\varphi_t + \nu_t \tag{3}$$

$$y_t = \begin{bmatrix} x_m \\ h_m \\ \psi m \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ h \\ \psi \end{bmatrix} + \nu_t \tag{4}$$

This system is nonlinear because of the state dynamics $\dot{\varphi}$, listed above in equation (2). To be precise, there is coupling between v, the control variable, and ψ, the internal state of the heading angle, in the state dynamics equation (2). The RLdMPC algorithm requires a linear model for control. To fit the RLdMPC framework, linearization of the simple unicycle model about the point (ψ_0, v_0, w_0) was attempted:

$$\dot{\varphi} = \dot{\varphi}_0 + \frac{\partial\varphi}{\partial\psi}\Delta\psi + \frac{\partial\varphi}{\partial v}\Delta v + \frac{\partial\varphi}{\partial\omega}\Delta\omega \tag{5}$$

Recall that a continuous linear state space model has the following form:

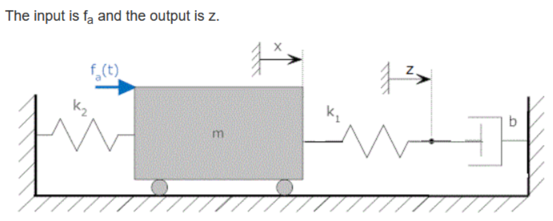

Fig. 3. Translational Dynamics

$$\dot{x} = A_c x + b_c u \tag{6}$$
$$y = \theta x \tag{7}$$

This unicycle linearization described in equation (5) can be worked further to get an expression in the following state space form:

$$\dot{\varphi} - \dot{\varphi}_0 = \begin{bmatrix} 0 & 0 & -v_0 sin\psi_0 \\ 0 & 0 & v_0 cos\psi_0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta h \\ \Delta \psi \end{bmatrix} + \begin{bmatrix} cos\psi_0 & 0 \\ sin\psi_0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta v \\ \Delta \omega \end{bmatrix} \tag{8}$$

In this case, the system is being linearized about some point (ψ_0, v_0, ω_0) where these values are the values of the state, and the control in the current time step. As time evolves, the values of (ψ_0, v_0, ω_0) may change depending on the state and control of the system at any given time step. The RLdMPC requires a constant A matrix for the linear system [1], which means the unicycle model may not be applied. As summarized in the paper [1] by Morinelly on page 1 equation (1), the system must be discrete and linear to match the following:

$$\varphi_{t+1} = A\varphi_t + bu_t \tag{9}$$
$$y_t = \theta\varphi_t + \nu_t \tag{10}$$

The unicycle problem formulation demonstrates one of the drawbacks of the RLdMPC algorithm: RLdMPC may not be able to accommodate nonlinear systems in its framework. To continue evaluation of the RLdMPC, a linear system must be formulated for analysis.

B. Linear Models

Initially, the RLdMPC method was applied to various linear dynamical systems to analyze the effects of the controller as it drives the system to steady state. In this section, we look at the translational dynamics of a system with a mass bounded by springs and a damper (Fig. 3), rotational dynamics of a system with two rotating masses connected by a damper and anchored by a spring (Fig. 4), and finally an RLC circuit with two inductors, a resistor, and a capacitor (Fig. 5). All of these systems take the continuous linear state space form described in equations (6) and (7).

In Fig. 3, the dynamics are represented by:

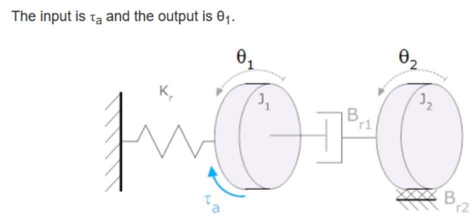

Fig. 4. Rotational Dynamics

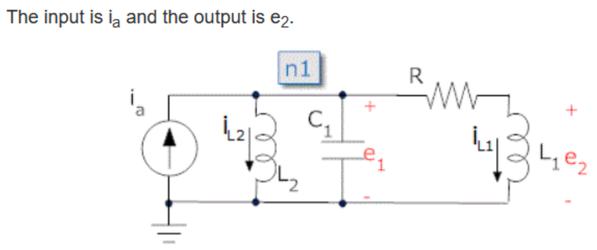

Fig. 5. Circuit Diagram

$$A_c = \begin{bmatrix} 0 & -1 & 0 \\ -(k_1+k_2)/m & 0 & k_1/m \\ k_1/b & 0 & -k_1/b \end{bmatrix}, \quad b_c = \begin{bmatrix} 0 \\ 1/m \\ 0 \end{bmatrix},$$

$$\theta = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

In Fig. 4, the dynamics are represented by:

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ -K_r/J_1 & -B_r1/J_1 & B_r1/J_1 \\ 0 & B_r1/J_2 & -(B_r1+B_r2)/J_2 \end{bmatrix},$$

$$b_c = \begin{bmatrix} 0 \\ -1/J_1 \\ 0 \end{bmatrix}, \quad \theta = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

In Fig. 5, the circuit state space is represented by:

$$A_c = \begin{bmatrix} 0 & 1/L_2 & 0 \\ -1/C_1 & 0 & -1/C_1 \\ 0 & 1/L_1 & -R/L_1 \end{bmatrix}, \quad b_c = \begin{bmatrix} 0 \\ 1/C_1 \\ 0 \end{bmatrix},$$

$$\theta_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \theta_2 = \begin{bmatrix} 0 \\ 0 \\ -R \end{bmatrix}$$

The RLC circuit was run twice, once with θ_1 (Fig. 10) using the voltage observation across the capacitor, and once with θ_2 (Fig. 11) using the current observation across inductor 1.

C. Discretization

The current definitions for the systems described in section B do not match the required form for the RLdMPC prescribed by equations (9) and (10). The systems in section B are all linear, but they are continuous and must be discretized. This discretization of the systems is straight forward for continuous linear time-invariant (LTI) systems, and can be achieved by the following transformation:

$$A = A_d = A_c dt + I \tag{11}$$
$$b = b_d = b_c dt \tag{12}$$
$$\theta = \theta_d = \theta_c \tag{13}$$

D. Quadratic Programming

The RLdMPC approach attempts to solve an optimization problem can be classified under a class of problems called the Quadratically Constrained Quadratic Program (QCQP) optimization problems. The QCQP class of optimization problems attempts to minimize quadratic functions subject to quadratic constraints. The general QCQP optimization problem takes the following form:

$$\begin{aligned} \min_x \quad & x^\intercal M_0 x + a_0^\intercal x \\ \text{s.t.} \quad & x^\intercal M_1 x + a_1^\intercal x = b_1 \\ & x^\intercal M_2 x + a_2^\intercal x = b_2 \\ & x_{min} \le x \le x_{max} \end{aligned} \tag{14}$$

The optimization problem needed to perform RLdMPC is described in page 5 equation (33) of Morinelly's paper [1], and it is the following:

$$\begin{aligned} \min_{u_N} \quad & \sum_{k=0}^{N-1} \alpha^k (\hat{y}_{t+k|t}^2 + r u_{t+k}^2 + \varphi_{t+k|t}^\intercal z_{t+k}) \\ & + \alpha^N \varphi_{t+N|t}^\intercal \hat{K}_t^* \varphi_{t+N|t} \\ \text{s.t.} \quad & \varphi_{t+k+1|t} = A\varphi_{t+k|t} + b u_{t+k} \\ & \hat{y}_{t+k|t} = \begin{cases} y_t, & k = 0 \\ \hat{\theta}_t^\intercal \varphi_{t+k|t}, & k > 0 \end{cases} \\ & R_{t+k} = \begin{cases} P_t^- 1, & k = 0 \\ R_{t+k-1} + \varphi_{t+k|t}\varphi_{t+k|t}^\intercal & k > 0 \end{cases} \\ & R_{t+k} z_{t+k} = \varphi_{t+k|t} \\ & \varphi_{t+k|t}^\intercal z_{t+k} \ge 0 \\ & y_{min} \le y_{t+\hat{k}|t} \le y_{max} \\ & u_{min} \le u_{t+\hat{k}|t} \le u_{max} \\ \text{for} \quad & k \in [0, N-1] \\ \text{given} \quad & \varphi_t, y_t, \hat{\theta}_t, P_t, \text{and } \hat{K}_t^* \end{aligned} \tag{15}$$

This attempts to minimize the finite horizon cost and the terminal cost by using the control vector $u_t, u_{t+1}, ..., u_{t+N-1}$. The minimization problem described in equation (15) can be formulated as a quadratic constraints problem described in equation (14). For the RLdMPC algorithm, the advantage of doing this is that it would allow the use of QCQP solvers to arrive at the solution. The BARON solver is used by Morinelly in her paper [1] as described in the paper's results section. In this paper, we attempt to use Gurobi solver via the Gurobi/MATLAB interface.

As described in Morinelly's paper [1], the minimization done by RLdMPC in equation (15) can be done by setting x to be, as :

$$x := \begin{bmatrix} \begin{bmatrix} y_{\hat{t}|t} & \dots & y_{t+\hat{N}-1|t} \end{bmatrix}^\intercal \\ \begin{bmatrix} u_t & \dots & u_{t+N-1} \end{bmatrix}^\intercal \\ \begin{bmatrix} \varphi_{t+1|t}^\intercal & \dots & \varphi_{t+N|t}^\intercal \end{bmatrix}^\intercal \\ \begin{bmatrix} z_t & \dots & z_{t+N-1} \end{bmatrix}^\intercal \\ \begin{bmatrix} \bar{R}_t & \dots & R_{t+\bar{N}-1} \end{bmatrix}^\intercal \end{bmatrix} \tag{16}$$

By doing this, (15) can be put into the QCQP form described by (14). The minimization problem can be specified by M_0 and a_0, the quadratic constraints by M_1 and M_2, and the linear constraints by a_1 and a_2.

E. Receding Horizon Control

A receding horizon approach is used to form the basis of the RLdMPC algorithm. In this manner, the controller may compute the observability of the system $\{A, \theta\}$ at each measurement step for the discretized system

$$\varphi_{t+1} = A\varphi_t + b u_t$$
$$y_t = \theta\varphi_t + \nu_t$$

where A and b represent the system and control matrices, respectively. The dynamics of this system may then be propagated to the next measurement time by estimating the hyper-state of the system. The hyper-state is defined in [5] as the augmented state

$$\mathbf{x} = \begin{bmatrix} \varphi \\ \hat{\theta} \end{bmatrix}$$

which allows for the joint state and parameter estimation of the true values. For this problem, the output matrix diagonal terms are estimated due to sensor noise at each measurement step. The output matrix estimates the true values in order to propagate a reasonable output prediction from which to design a controller input. The suboptimal control chosen for the next time step will be dependent on the output estimate. This anticipated information dictating the decision made in the future makes the control method a certainty equivalence control.

F. Estimation Theory

From the formulation of the RLdMPC outlined in the paper by Heirung et al. [8], Morinelly and Ydstie [1] formulate a general dual MPC control algorithm that includes anticipated information which provides the agent with the ability to introduce exploration in optimal decisions along the control path. With this anticipation, captured through the output parameters θ, a set of equations are derived that are used to implement

```
Initialization: Specify the initial state $\varphi_{t0}$, output
parameter estimate, $\theta_{t_0}$ and its covariance matrix,
$P_{t_0}$. Set $\hat{K}_{t_0} = 0_n$. Set $t_0 \to t$

Step 1: Verify observability of $\{A, \hat{\theta}\}$. Since $\hat{\theta}$ is estimated
at each measurement step, the system
may become unobservable and thus,
the discrete ricatti equation solving for $\hat{K}^*$
may not be feasible. If not feasible,
use the previous $\hat{K}^*$
value.

Step 2: Estimate the measurement using an MPC
model with finite horizon, N and minimizing (14)

Step 3:Implement the $u_t$ to move agent
Take another measurement and
update estimation parameters
```

Fig. 6. RLdMPC pseudo-code

a Certainty Equivalence strategy for the RLdMPC. The state transitions are defined as:

$$\varphi_{t+k+1|t} = A\varphi_{t+k|t} + bu_{t+k}$$

The error covariance matrix associated with the output estimate, $\hat{\theta}$

$$P_t := E[(\theta - \hat{\theta}_t)(\theta - \hat{\theta}_t)^T|Y_t]$$

The parameter uncertainty, R, is propagated forward in time with the Recursive Least Squares:

$$R_{t+k+1} = R_{t+k} + \varphi_{t+k+1|t}\varphi_{t+k+1|t}^T$$

The nonlinear cost function:

$$\hat{c}_{t+k|t} := \hat{y}_{t+k|t}^2 + ru_{t+k}^2 + \varphi_{t+k+1|t}^T P_{t+k|t}\varphi_{t+k+1|t}$$

can be expressed in terms of the parameter uncertainty with the introduction of the variable, z, giving the following:

$$R_{t+k}z_{t+k} = \varphi_{t+k|t}$$

This gives a final cost function of:

$$\hat{c}_{t+k|t} := \hat{y}_{t+k|t}^2 + ru_{t+k}^2 + \varphi_{t+k+1|t}^T z_{t+k}$$

This stage cost function gives an output predictor that can be computed exactly with these equations above.

III. Problem Analysis

A. Policy Cost Analysis

For the rotational dynamics system described in the previous section, The following output feedback regulator methods were attempted:

1) optimal policy, μ generated from

$$\mu(\varphi)* = -(b^T K^* b + r)^{-1} b^T K^* A\varphi$$

where K^* denotes the optimal cost matrix of the system found through solving the ricatti equation

$$K^* = A^T(K^* - K^*b(b^T K^* b + r)^{-1} b^T K^*)A + \hat{\theta}\hat{\theta}^T$$

2) RLdMPC: N=1

The equations above are used from the original RLdMPC paper.

A comparison is illustrated by Figure 7 and 9. The results in Figure 7 were obtained assuming perfect knowledge of the rotating system in Fig. 4 for a finite horizon problem. The plot of the system output shows that there is no learning occurring while the agent is only controlling the states with no anticipation of the output. On the contrary, 9 shows a smooth arrival to steady state while allowing the system to explore but still meeting the desired goal.

Alternatively, a more robust solution can be used by approaching this problem using the quadratic constraints of the cost function in section 2. This allows us to minimize both the input and output when analyzing the objective function, results can be seen in Figure 8 - 11. The benefit of this method settles after half of the time as the MPC approach.

B. Applications to Dynamical and Electrical Systems

Here we see the results of applying the RLdMPC with a horizon of N = 1 to the various dynamical and electrical systems described in section B of the Problem Formulation. In all the cases, we see the RLdMPC drive the system state to zero. As the bounds for the control input were explored, it was seen that over the first time steps, the initial control input would spike to near the upper or lower bound. However, as that bound was decreased, there was no appreciable increase in time for the system to go to zero.

From the circuit results, Fig. 10 and Fig. 11, we see that the RLdMPC provides similar responses for both cases, but with different control inputs at each time step. We see that in both cases, the states are being driven to zero, however due to the fact that the horizon is so small in this RLdMPC implementation, the cost function being minimized leads to a system input that eventually drives the states to zero, but with the smallest input possible for each iteration. To potentially speed up the process and have the system get driven to zero faster, larger horizons would provide a more efficient control input to send the system states to zero in less time.

C. RLdMPC with finite horizon $N > 1$

After successfully implementing RLdMPC for $N = 1$ on linear dynamical and electrical systems, we attempt to control those same linear systems for finite horizon length $N = 2$.

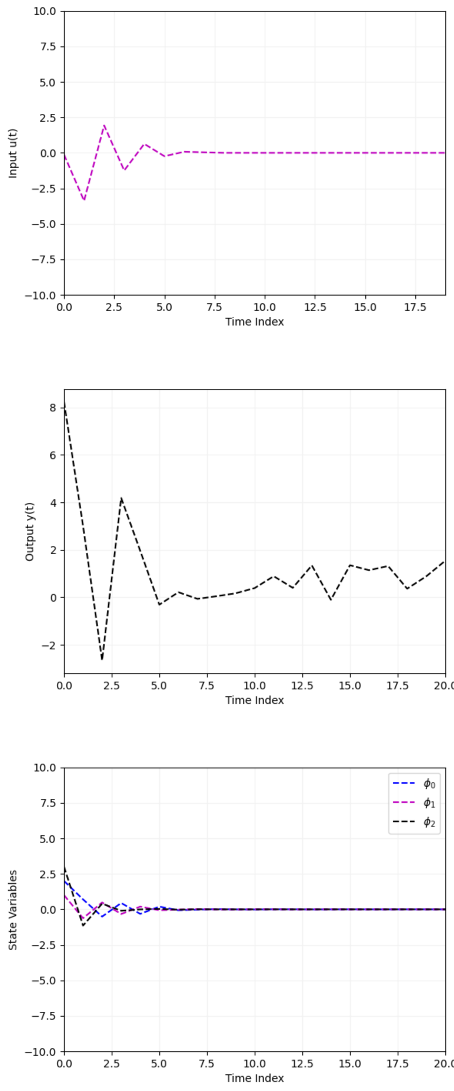

Fig. 7. Input Variable (top), Uncontrolled Variable(mid), Controlled Variables(bottom) when using finite horizon MPC model

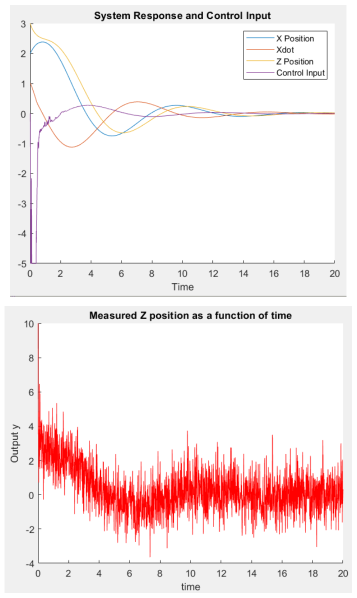

Fig. 8. System Response (top), Measurement of System (bottom) when solving translational system

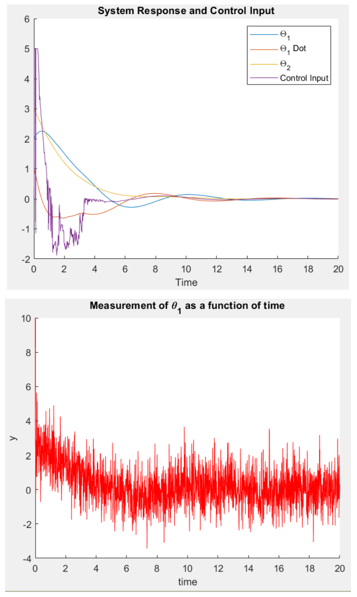

Fig. 9. System Response (top), Measurement of System (bottom) when solving rotational system

This required the implementation of the optimization problem described in equation (15) for $N = 2$ by using the QCQP solver , Gurobi. For $N = 2$, there were significantly more quadratic constraints, which made the problem more computationally intensive. Morinelly's paper [1] describes specifically that there are $n^2 + N(n + 1)$ linear equality constraints and $(N - 1)n^2 + Nn$ quadratic equality constraints, where $n = \dim(\varphi)$ and N is the length of the finite horizon. As the finite horizon increases, quadratic constraints increase linearly. This is the reason why we were faced with significantly more constraints when trying to solve equation for finite horizon length $N = 2$(15). An implementation for $N = 2$ can be found in the accompanying code "dMPC_gurobi_n2.m". The implementation matched the number of linear and quadratic constraints that Morinelly's paper predicts for $N = 2$. However, the solution failed to converge due to the complexity of the problem. This showed us that equation (15) may need either specific tolerances, or special solvers such as BARON, to converge via the QCQP approach. This highlights the computational cost associated with implementing RLdMPC.

D. Summary

Although the RLdMPC approach is computationally expensive due to it being a quadratically constrained quadratic program of high dimensionality, with additional nonlinear constraints as the horizon increases, it balances the ability of the agent to explore and meet a goal while the critic increases its certainty as it predicts the optimal policy. Injecting uncertainty

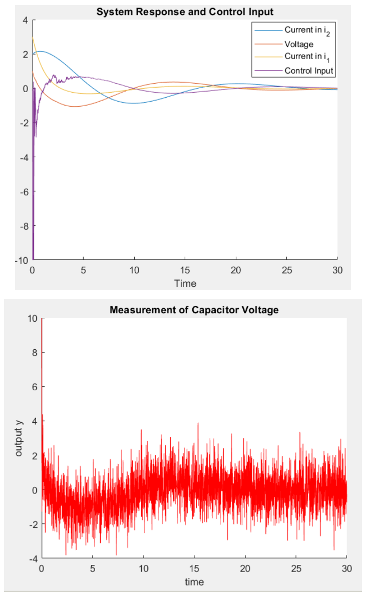

Fig. 10. System Response (top), Measurement of System (bottom) when solving electrical system and observing voltage

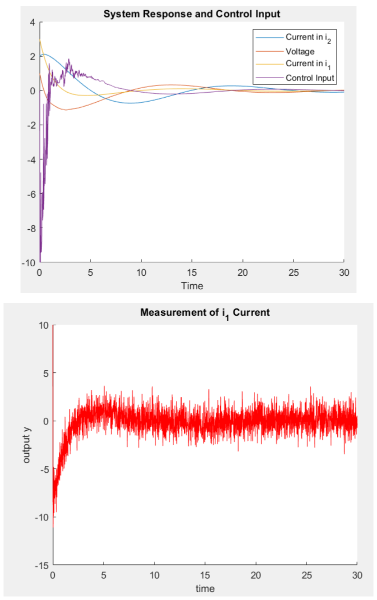

Fig. 11. System Response (top), Measurement of System (bottom) when solving electrical system and observing current

into the system proves to allow the agent to learn the optimal input that can minimize its objective. The addition of state estimation also allows the controller to operate with less-than ideal systems or real systems to provide near-optimal control policies, even in the presence of noise which would otherwise render similar optimal controllers, like the linear-quadratic regulator, unusable. This provides a new facet of model predictive control that provides additional capabilities to operate in the real world conditions.

E. Future Work

In the future, it may be beneficial investigating adjustments to the RLdMPC algorithm to fit non-linear MIMO systems like the unicycle model, and more complex real world systems.

F. Code

For further references to this project see GITHUB

References

[1] Morinelly, Juan E., and B. Erik Ydstie. “Dual MPC with Reinforcement Learning.” IFAC-PapersOnLine, Elsevier, 9 Aug. 2016, PDF.

[2] Warren, Powell B. What You Should Know about Approximate Dynamic Programming, Wiley Science, 24 Feb. 2009, PDF.

[3] Bertsekas, Dimitri P. “Approximate Dynamic Programming.” CEA - Cadarach France. 2012.

[4] Wittenmark, Bjorn ”Adaptive dual control methods: an overview.” *In 5th IFAC Symposium on Adaptive Systems in Control and Signal Processing*

[5] B. Wittenmark, “Adaptive dual control,” in Control Systems, Robotics and Automation, Encyclopedia of Life Support Systems (EOLSS), Devel-oped Under the Auspices of the UNESCO. Oxford, U.K.: Eolss Pub.,Jan. 2002

[6] Filatov, N. M., and Unbehauen, H. (2004). Adaptive dual control: Theory and applications. Berlin: Springer.

[7] A.A. Fel’dbaum, Dual control theory. I, Automation Remote Control 21 (1960) 874–880;
A.A. Fel’dbaum, Dual control theory. II, Automation Remote Control 21 (1960) 1453–1464;
A.A. Fel’dbaum, Dual control theory. III, Automation Remote Control 22 (1961) 1–12;
A.A. Fel’dbaum, Dual control theory. IV, Automation Remote Control 22 (1961) 109–121.

[8] Tor Aksel N. Heirung, B. Erik Ydstie, Bjarne Foss, Dual MPC for FIR Systems: Information Anticipation, IFAC-PapersOnLine, Volume 48, Issue 8, 2015, Pages 1033-1038, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2015.09.104.

[9] Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. Ph.D. thesis, University of Cambridge England.

[10] Ng, A.Y.: Shaping and policy search in reinforcement learning. PhD thesis, University of California, Berkeley (2003)

[11] Gaudet, Brian, et al. “Deep Reinforcement Learning for Six Degree-of-Freedom Planetary Landing.” Advances in Space Research, Pergamon, 7 Jan. 2020

[12] Gaudet, Brian, et al. “Reinforcement Learning for Angle-Only Intercept Guidance of Maneuvering Targets.” Aerospace Science and Technology, Elsevier Masson, 30 Jan. 2020

[13] Chen, Yujiao, et al. “Optimal Control of HVAC and Window Systems for Natural Ventilation through Reinforcement Learning.” Energy and Buildings, Elsevier, 29 Mar. 2018

[14] El-Fakdi, A., et al. “Direct Policy Search Reinforcement Learning for Autonomous Underwater Cable Tracking.” IFAC Proceedings Volumes, Elsevier, 21 Apr. 2016

[15] S. Gu, E. Holly, T. Lillicrap and S. Levine, ”Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017