# Inverted Pendulum Stabilization via Learning-Based MPC

Brendon Anderson Lua Varner

10 May 2019

## 1 Introduction

The control of a dynamical system is motivated by two critical (and often conflicting) problems: (i) to ensure robustness and stability of the system's dynamic response, and (ii) to optimize the performance of the response. Control strategies such as linear quadratic regulation (LQR) and model predictive control (MPC) have become popular due to their rich theoretical foundations and their robustness and stability guarantees, but their formulations typically revolve around linear, time-invariant (LTI) approximations of the true system dynamics. Revitalized interest in improving closed-loop performance while maintaining these robustness and stability guarantees has led to the incorporation of statistical and machine learning techniques into the control design procedure. Learning-based model predictive control (LBMPC) provides a methodical formulation of this novel approach [4].

The underlying concept behind LBMPC is to use two decoupled models of the system dynamics during the control design process. The first model is a simple LTI approximation of the system dynamics which is used to derive deterministic guarantees on the robustness and stability properties of the resulting closed-loop system. The second model, novel in the LBMPC formulation, is an adaptive model of the system which learns the true underlying dynamics of the system over time. Choosing control inputs according to the learned (more accurate) dynamics, yet subject to the robustness constraints on the first model, results in lower cost and higher performance [4].

In this report, we perform an empirical study to verify the proposed efficacy of learning-based model predictive control. These simulations are carried out on two systems ubiquitous within the academic control community: (i) a simple inverted pendulum, and (ii) an inverted pendulum on a cart. Through numerical experiments, we demonstrate some of the theoretical guarantees inherent to LBMPC and demonstrate when LBMPC offers performance advantages over the more classic LQR and (robust) MPC control strategies.

## 2 Theory

Suppose the underlying dynamics of a discrete-time dynamical system are given by

$$x(t+1) = f(x(t), u(t)) + d(t), \tag{1}$$

where $f\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a continuously differentiable (and possibly nonlinear) function, and $d \in \mathbb{R}^n$ is additive disturbance. Without loss of generality, we assume $f(0,0) = 0$ (this can always be made true by a change of coordinates). Our goal is to design a control input, $u \in \mathbb{R}^m$, to drive the state of the system, $x \in \mathbb{R}^n$, to the equilibrium point $(x_e, u_e) = (0, 0)$. In general, this equilibrium point may be unstable, indicating that convergence to equilibrium requires an intelligent choice for the input $u$. Furthermore, our choice of $u$ should be robust in the sense that our design goals (such as convergence to equilibrium, and possibly constraints on $x$ and $u$) should remain satisfied in the presence of $d$. In this work we consider three strategies for choosing the control input $u$: linear quadratic regulation (LQR), linear model predictive control (MPC), and learning-based model predictive control (LBMPC).

The three control strategies considered here share a few common features. First, they utilize a linear time-invariant (LTI) model to approximate the system dynamics as $x(t+1) = Ax(t) + Bu(t)$. [At least in this report we will restrict ourselves to the use of LTI models as constraints on the system dynamics used in the optimization problems that follow. Extensions beyond LTI models may be useful, but are not considered here.] This model can be found by using Jacobian linearization of $f$ about the equilibrium point $(x_e, u_e)$:

$$A = \nabla_x f(x_e, u_e)^\top = \left.\begin{bmatrix} \frac{\partial f_1(x,u)}{\partial x_1} & \frac{\partial f_1(x,u)}{\partial x_2} & \cdots & \frac{\partial f_1(x,u)}{\partial x_n} \\ \frac{\partial f_2(x,u)}{\partial x_1} & \frac{\partial f_2(x,u)}{\partial x_2} & \cdots & \frac{\partial f_2(x,u)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x,u)}{\partial x_1} & \frac{\partial f_n(x,u)}{\partial x_2} & \cdots & \frac{\partial f_n(x,u)}{\partial x_n} \end{bmatrix}\right|_{(x,u)=(x_e,u_e)},$$

$$B = \nabla_u f(x_e, u_e)^\top = \left.\begin{bmatrix} \frac{\partial f_1(x,u)}{\partial u_1} & \frac{\partial f_1(x,u)}{\partial u_2} & \cdots & \frac{\partial f_1(x,u)}{\partial u_m} \\ \frac{\partial f_2(x,u)}{\partial u_1} & \frac{\partial f_2(x,u)}{\partial u_2} & \cdots & \frac{\partial f_2(x,u)}{\partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(x,u)}{\partial u_1} & \frac{\partial f_n(x,u)}{\partial u_2} & \cdots & \frac{\partial f_n(x,u)}{\partial u_m} \end{bmatrix}\right|_{(x,u)=(x_e,u_e)}.$$

The next common trait of the three control strategies of interest is that they each can be expressed as mathematical optimization problems. In each case, objective functions $\psi_t \colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $t \in \mathbb{Z}_+$, are to be minimized. The design of these functions plays a crucial role in how the resulting control system performs, as well as the tractability of the optimization problems at hand. In general, the functions $\psi_t$ can vary at each time step. Regardless, they should be interpretable as a measure of the deviation of the state and input trajectories, $x(t)$ and $u(t)$, from the target equilibrium point $(x_e, u_e)$. In this report, we take the objective functions to be (strongly) convex quadratics, which allows for easy tuning of the relative importance of certain states and inputs in the overall performance, in addition to providing unique global optimality guarantees for the resulting solution $u^*$.

The other common characteristics and parameters will be immediately clear after introducing each technique. We will begin with LQR control.

## 2.1 Linear Quadratic Regulator

The linear quadratic regulator (LQR) strategy solves the following optimization problem at the starting time $t = 0$:

$$\begin{aligned} \text{minimize} \quad & \sum_{t=0}^{\infty} (x(t)^\top Q x(t) + u(t)^\top R u(t)) \\ \text{subject to} \quad & x(0) = \xi, \\ & x(t+1) = Ax(t) + Bu(t), \ \forall t \in \mathbb{Z}_+, \end{aligned} \tag{2}$$

with optimization variables $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ for $t \in \mathbb{Z}_+$. Here, we denote the initial condition of the system by $\xi \in \mathbb{R}^n$. The problem has two tuning parameters, $Q \succeq 0$ and $R \succ 0$, which penalize the state and input trajectories, respectively. In the case that $(A, B)$ is stabilizable, the unique solution to (2) is given by $u^*$ satisfying

$$\begin{aligned} u^*(t) &= Kx(t), \\ K &= -(R + B^\top PB)^{-1}B^\top PA, \\ P &= Q + A^\top PA - A^\top PB(R + B^\top PB)^{-1}B^\top PA, \end{aligned}$$

where $P \succeq 0$ [6]. Note that the state trajectory $x$ is uniquely determined by $u^*$ in this case. Furthermore, the closed-loop system $x(t+1) = (A + BK)x(t)$ is guaranteed to be stable at the equilibrium point of interest $(x_e, u_e) = (0, 0)$.

## 2.2 Linear Model Predictive Control

One of the pitfalls of LQR control, as given by (2), is that constraints on the state trajectory $x$ and the control input $u$ cannot directly be taken into account without surrendering the elegant solution stated in Section 2.1. Furthermore, in LQR the optimal control input $u^*$ is determined one time at the system's initiation. What if, at some point during the closed-loop response the system is subjected to unforeseen disturbances, or the actual state trajectory begins to deviate from the precomputed optimal trajectory $x(t) = (A + BK)^t x(0)$? In this case, LQR control has no way to adjust the control action on-the-fly. With model predictive control, we introduce a method to handle state and input constraints explicitly, as well as a means to update the optimal control input in a real-time fashion as new state measurements are taken [5].

Let us start by rewriting (1) as $x(t+1) = Ax(t) + Bu(t) + g(x(t), u(t))$ where $g(x(t), u(t)) = f(x(t), u(t)) + d(t) - Ax(t) - Bu(t)$. Now suppose that the term associated with the system's nonlinearity and disturbance has a known bound: $g(x, u) \in \mathcal{W} \subset \mathbb{R}^n$, $\forall x \in \mathcal{X}$, $u \in \mathcal{U}$. The sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are constraint sets in which the true state and input trajectories are to reside; $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$ for all $t \in \mathbb{Z}_+$. Typically, domain specific knowledge is used to design these constraint sets. Then, the (robust) linear model predictive control (MPC) strategy solves the following optimization problem at each time step $t \in \mathbb{Z}_+$:

$$\begin{array}{ll} \text{minimize} & \bar{x}(t+N)^\top P\bar{x}(t+N) + \displaystyle\sum_{k=t}^{t+N-1} (\bar{x}(k)^\top Q\bar{x}(k) + \check{u}(k)^\top R\check{u}(k)) \\ \text{subject to} & \bar{x}(t) = x(t), \\ & \bar{x}(t+k+1) = A\bar{x}(t+k) + B\check{u}(t+k), \\ & \check{u}(t+k) = K\bar{x}(t+k) + c(t+k), \\ & \bar{x}(t+k+1) \in \mathcal{X} \ominus \mathcal{R}_{k+1}, \\ & \bar{x}(t+N) \in \Omega \ominus \mathcal{R}_N, \\ & \check{u}(t+k) \in \mathcal{U} \ominus (K\mathcal{R}_k), \ \forall k \in \{0, 1, \ldots, N-1\}, \end{array} \tag{3}$$

with optimization variables $\bar{x}(t+k)$ for $k \in \{0, 1, \ldots, N\}$, as well as $\check{u}(t+k)$ and $c(t+k)$ for $k \in \{0, 1, \ldots, N-1\}$. Here, $P \in \mathbb{S}^n_+$ and $K \in \mathbb{R}^{m \times n}$ are taken to be the solution to the infinite horizon LQR problem, as described in Section 2.1. The control invariant set (also called terminal set) $\Omega \subseteq \mathbb{R}^n$ is where we require the state to reside at the end of the prediction horizon; $x(t+N) \in \Omega$ for each prediction step $t \in \mathbb{Z}_+$. It can be shown that the maximal terminal set which guarantees recursive feasibility and constraint satisfaction of the problem (3) at each time step $t \in \mathbb{Z}_+$ satisfies: (i) constraint satisfaction, $\Omega \subseteq \{x : x \in \mathcal{X},\ Kx \in \mathcal{U}\}$, and (ii) disturbance invariance, $(A + BK)\Omega \oplus \mathcal{W} \subseteq \Omega$ [1]. Finally, the disturbance tubes are computed as $\mathcal{R}_0 =$

$\{0\},\ \mathcal{R}_k = \oplus_{j=0}^{k-1}(A+BK)^k\mathcal{W},\ k \in \{1,2,\dots,N\}$. Intuitively, these disturbance tubes are used to subtract out the propagation of nonlinearity and disturbance from the predicted state and input trajectories, ensuring the actual state and input trajectories satisfy their respective constraints in a robust manner.

After solving the above finite horizon optimal control problem, the first computed optimal control input is sent to the system, i.e. $u(t) = \check{u}^*(t)$, and the process is repeated at each of the time steps $t+1, t+2, \dots$ that follow.

## 2.3 Learning-Based Model Predictive Control

One downside to MPC, as given by (3), is that the model used in the optimization problem to represent the system's dynamics is constant at each time step. This is seemingly wasteful, since new state response measurements are stored at each time step, and therefore these measurements have the capability of providing adaptation to the model used. This is the underlying methodology behind learning-based MPC: use a constant LTI model to predict the system's response alongside the robustified constraints described in Section 2.2, which guarantees robust constraint satisfaction, and use a separate model which adaptively approximates the true (nonlinear) system dynamics as new data becomes available, which yields more accurate state and input predictions to use in the objective function.

The learning-based model predictive control (LBMPC) strategy solves the following optimization problem at each time step $t \in \mathbb{Z}_+$:

$$\begin{aligned}
\text{minimize} \quad & \tilde{x}(t+N)^\top P\tilde{x}(t+N) + \sum_{k=t}^{t+N-1} \left(\tilde{x}(k)^\top Q\tilde{x}(k) + \check{u}(k)^\top R\check{u}(k)\right) \qquad (4)\\
\text{subject to} \quad & \tilde{x}(t) = x(t),\\
& \bar{x}(t) = x(t),\\
& \tilde{x}(t+k+1) = A\tilde{x}(t+k) + B\check{u}(t+k) + \mathcal{O}_t(\tilde{x}(t+k), \check{u}(t+k)),\\
& \bar{x}(t+k+1) = A\bar{x}(t+k) + B\check{u}(t+k),\\
& \check{u}(t+k) = K\bar{x}(t+k) + c(t+k),\\
& \bar{x}(t+k+1) \in \mathcal{X} \ominus \mathcal{R}_{k+1},\\
& \bar{x}(t+N) \in \Omega \ominus \mathcal{R}_N,\\
& \check{u}(t+k) \in \mathcal{U} \ominus (K\mathcal{R}_k),\ \forall k \in \{0,1,\dots,N-1\},
\end{aligned}$$

with optimization variables $\tilde{x}(t+k)$ and $\bar{x}(t+k)$ for $k \in \{0,1,\dots,N\}$, as well as $\check{u}(t+k)$ and $c(t+k)$ for $k \in \{0,1,\dots,N-1\}$. Here, in contrast to the linear MPC problem (3), we use two separate state prediction variables, $\tilde{x}$ and $\bar{x}$. The variable $\tilde{x}$ is used in the adaptive model and is intended to yield more accurate predictions for the actual state trajectory. This increased accuracy is a result of the oracle function $\mathcal{O}_t$, which varies with $t$, and iteratively approximates the nonlinear portion of the system's dynamics $g$. That is, as more state measurements become available, it is our goal that $\mathcal{O}_t \to g$. The estimation used to compute the oracle $\mathcal{O}_t$ is performed at each time step, and can typically be formulated as a least squares problem (this is further discussed in Section 3) [3]. The same recursive feasibility and constraint satisfaction guarantees hold for LBMPC as noted for the linear MPC case [2].

# 3 Results

## 3.1 Implementation

The results that follow were computed using Matlab, CVX, and MPT. We provide the scripts and functions we have written to obtain the following results here: `https://git.io/fjcAh`. CVX is available here: `http://cvxr.com/cvx/`. MPT is available here: `https://www.mpt3.org/`.

We use Matlab's `dlqr` function to solve the LQR problem, CVX to solve the remaining optimization problems, and MPT to perform polytope definitions, manipulations, and plotting. For ease of implementation, we restrict our constraint sets to be box constraints (discussed in more detail to come), which are simple to define using MPT. Additionally, we used augmented variables, for example $\bar{x} := (\bar{x}(t), \bar{x}(t+1), \dots, \bar{x}(t+N)) \in \mathbb{R}^{n(N+1)}$, in the CVX code. This removes the summations found in the objective functions of (3) and (4), and also allows us to write the state, input, and terminal set constraints more compactly in one line each.

In this report, we restrict ourselves to the use of a linear oracle, i.e.

$$\mathcal{O}_t(x,u) = H_t x + G_t u.$$

In order to compute the oracle at time $t$, we first compute the estimated nonlinearities, $y(k) = x(k+1) - (Ax(k) + Bu(k))$, which represent approximations of $g(x(k), u(k))$ at each previous time step $k \in \{0, 1, \dots, t\}$. Now, since we would like the oracle to converge to $g$, we minimize the sum of squared errors between the oracle and the approximate nonlinearities $y(k)$:

$$\text{minimize} \sum_{k=0}^{t} \| y(k) - (Hx(k) + Gu(k)) \|_2^2,$$

where the optimization variables are $H \in \mathbb{R}^{n\times n}$ and $G \in \mathbb{R}^{n\times m}$. If we define the augmented nonlinear approximation matrix $Y(t) = \begin{bmatrix} y(0) & y(1) & \cdots & y(t) \end{bmatrix}$, the augmented state matrix $X(t) = \begin{bmatrix} x(0) & x(1) & \cdots & x(t) \end{bmatrix}$, and the augmented input matrix $U(t) = \begin{bmatrix} u(0) & u(1) & \cdots & u(t) \end{bmatrix}$, then it is straightforward to show that the above optimization problem can be equivalently reformulated as

$$\text{minimize} \ \| Y(t) - HX(t) - GU(t) \|_F^2.$$

We use implement this problem in CVX to solve for the updated oracle parameters $H_t$ and $G_t$ at each time step of the LBMPC problem.

## 3.2 Simple Inverted Pendulum

The first system we will apply LBMPC to is the simple inverted pendulum shown in Figure 1. In this case, the input to the system is a torque applied at the pivot of the pendulum, denoted $\tau \in \mathbb{R}$. We denote the pendulum's angular position (from the upright) as $x_1 \in \mathbb{R}$, and allow for a viscous damping coefficient of $\beta \in \mathbb{R}_+$.

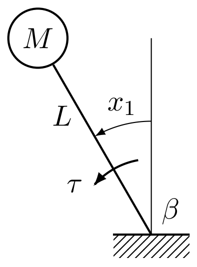

Figure 1: Simple inverted pendulum diagram.

### 3.2.1 Dynamic System Model

Applying Newton's second law, it is a simple exercise to show that the dynamics of this system are governed by

$$ML^2\ddot{x}_1 + \beta\dot{x}_1 - MLg\sin(x_1) = \tau,$$

where $M$ represents the pendulum mass, $L$ represents its length, and $g$ is the acceleration due to gravity. Defining $x_2 = \dot{x}_1$, we obtain the following nonlinear continuous-time state-space representation:

$$\dot{x} = \begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = f_{\text{ct}}(x,u) \coloneqq \begin{bmatrix} x_2 \\ -\frac{\beta}{ML^2}x_2 + \frac{g}{L}\sin(x_1) + u\end{bmatrix},$$

where we have defined the scaled input to be $u = \frac{1}{ML^2}\tau$. Using the first-order forward Euler method, we can discretize the system as $x(t+1) \approx Tf_{\text{ct}}(x(t), u(t)) + x(t)$, where $T$ is the sampling time. We use this approach and define the resulting discrete-time system as our "ground truth" system with which to run our experiments. We also assume there is random disturbances $d \in \mathbb{R}^2$, uniformly distributed on the box $\mathcal{D} = [d_{\min}, d_{\max}]^2$. With this, we obtain the following as our ground truth system:

$$x(t+1) = f(x(t), u(t)) + d(t) = \begin{bmatrix} x_1(t) + Tx_2(t) \\ (1 - \frac{\beta T}{ML^2})x_2(t) + \frac{gT}{L}\sin(x_1(t)) + Tu(t)\end{bmatrix} + d(t). \tag{5}$$

As described in Section 2, the control methods of interest in this report require an LTI model of the system which can be obtained through Jacobian linearization. Applying this approach to (5), we obtain the following model linearized about $(x_e, u_e) = (0, 0)$:

$$x(t+1) \approx Ax(t) + Bu(t) = \begin{bmatrix} 1 & T \\ \frac{gT}{L} & 1 - \frac{\beta T}{ML^2}\end{bmatrix} x(t) + \begin{bmatrix} 0 \\ T \end{bmatrix} u(t). \tag{6}$$

### 3.2.2 Constraint Sets

The state and input constraints are taken to be box constraints for simplicity: $\mathcal{X} = [x_{\min}, x_{\max}]^2$ and $\mathcal{U} = [u_{\min}, u_{\max}]$. As mentioned above, we assume the random disturbances lie within the box constraint $\mathcal{D} = [d_{\min}, d_{\max}]^2$. Now, under the (mild) assumptions that $0 \leq x_1(t) \leq x_1(0)$ and $x_1(t) - \sin(x_1(t)) \leq \sin(x_1(0))$ for all $t \in \mathbb{Z}_+$, it can be easily shown that $\frac{gT}{L}(\sin(x_1(t)) - x_1(t)) \in [-\frac{gT}{L}\sin(x_1(0)), \frac{gT}{L}\sin(x_1(0))]$, $\forall t \in \mathbb{Z}_+$. Therefore, we find that

$$\begin{bmatrix} 0 \\ \frac{gT}{L}(\sin(x_1(t)) - x_1(t))\end{bmatrix} \in \mathcal{E} \coloneqq 0 \times \left[-\frac{gT}{L}\sin(x_1(0)), \frac{gT}{L}\sin(x_1(0))\right].$$

Hence, since $g(x(t), u(t)) = (0, \frac{gT}{L}(\sin(x_1(t)) - x_1(t))) + d(t)$, we find the following polytope bound on the overall disturbance:

$$g(x(t), u(t)) \in \mathcal{W} = \mathcal{D} \oplus \mathcal{E} = [d_{\min}, d_{\max}]^2 \oplus \left(0 \times \left[-\frac{gT}{L}\sin(x_1(0)), \frac{gT}{L}\sin(x_1(0))\right]\right). \tag{7}$$

With these constraint polytopes computed, we are in a place to implement LQR, (robust) MPC, and LBMPC. We perform three separate experiments to showcase different features of these control strategies, as outlined in the next sections.

### 3.2.3 Experiment #1: Stabilization

We begin with the following setup: $T = 0.5$, $M = 1$, $L = 1$, $\beta = 0.01$, $g = 10$, $N = 5$, $[d_{\min}, d_{\max}] = [-0.01, 0.01]$, $[x_{\min}, x_{\max}] = [-1, 1]$, $[u_{\min}, u_{\max}] = [-10, 10]$, $Q = I_n = I_2$, and $R = I_m = 1$. We begin by initializing the system to $x(0) = 0$ and allow the disturbances to perturb the state away from this equilibrium point. Without control, we see that the system is unstable, as shown in Figure 2.

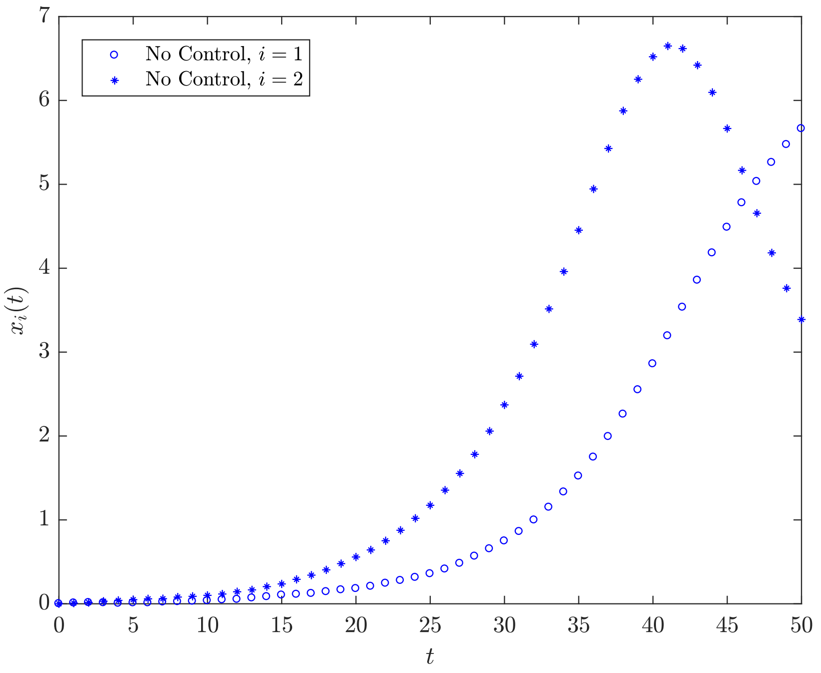

Figure 2: Inverted pendulum without control input. The trajectories shown depict the instability of the equilibrium point $x = 0$.

Next, we set $x(0) = (0.5, 0)$. By running LQR, linear (robust) MPC, and LBMPC, we find that the system is successfully stabilized to the origin by all three strategies, as shown in Figure 3. None of the state or input constraints are active, and therefore we find similar performance between the three methods. We plot the constraint and disturbance polytopes in Figure 4 to visualize this assertion. We also display the control invariant set $\Omega$ (as computed by MPT) in Figure 5. This is the target set into which MPC and LBMPC aim to land the state trajectory to guarantee recursive feasibility and constraint satisfaction. Finally, we plot the disturbance tubes, $\mathcal{R}_0 = \{0\}$, $\mathcal{R}_k = \oplus_{j=0}^{k-1}(A + BK)^k\mathcal{W}$, $k \in \{1, 2, \dots, N\}$, in Figure 6. As expected, the tubes grow as the disturbance polytope $\mathcal{W}$ is propogated through the system dynamics at each time step. Furthermore, we find that $\mathcal{R}_{k+1} \subset \mathcal{X}$ for each $k\{1, 2, \dots, N-1\}$, indicating that the robustified state constraint set $\mathcal{X} \ominus \mathcal{R}_k$ is nonempty, yielding feasible polytope constraints in the MPC and LBMPC optimization problems. Similar assertions are seen to hold for the robustified input constraint sets.

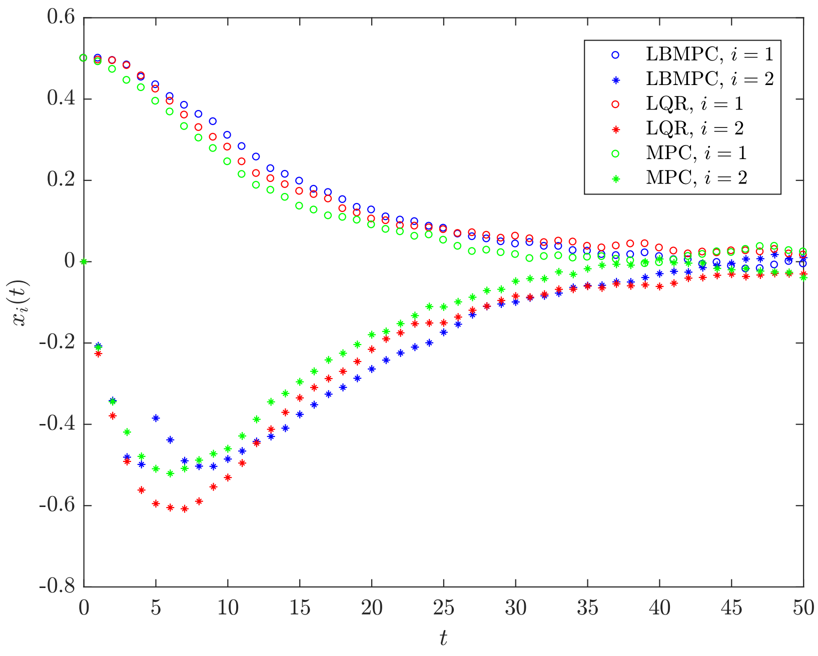

Figure 3: Inverted pendulum with control input. LBMPC is seen to stabilize the system with performance on par with that of LQR and linear MPC.

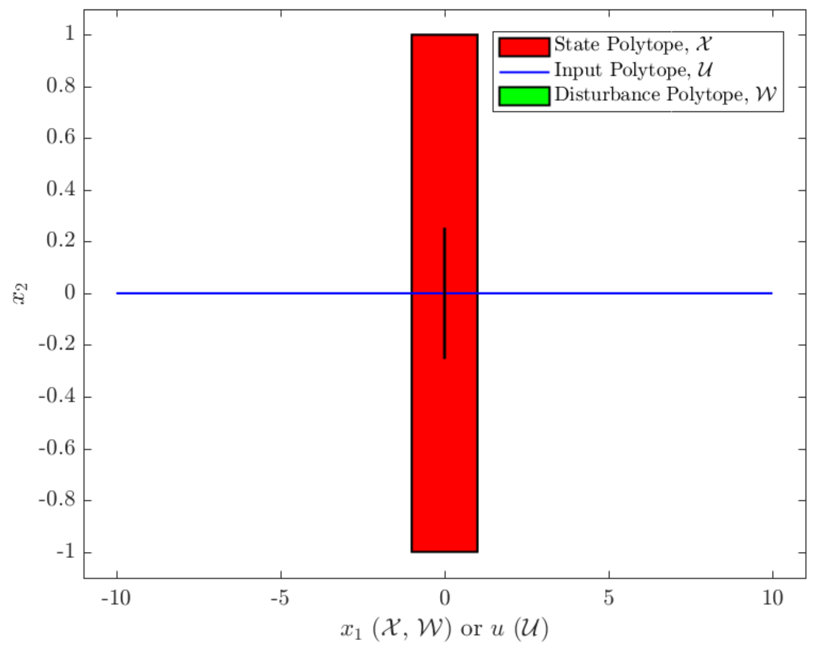

Figure 4: Constraint and disturbance polytopes for use in (robust) MPC and LBMPC.

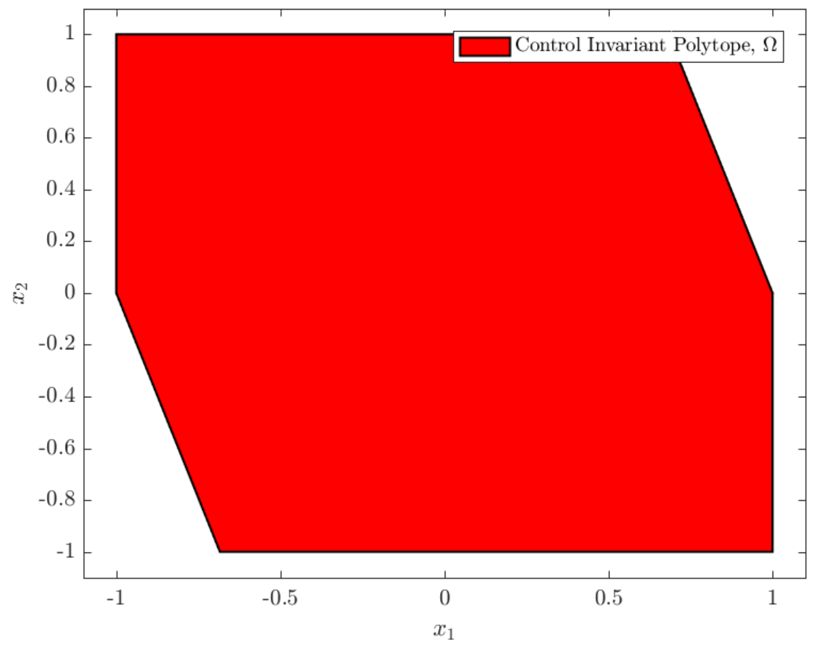

Figure 5: Control invariant set for use in (robust) MPC and LBMPC.

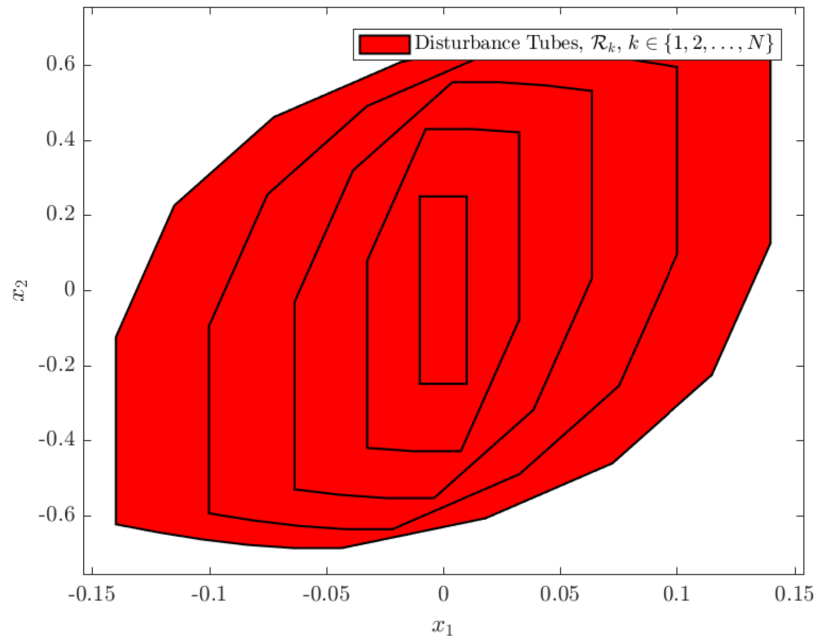

Figure 6: Disturbance tubes at each step of the MPC horizon. As expected, the disturbance tubes grow with the dynamics of the system.

### 3.2.4 Experiment #2: Constraint Saturation

We now lower the initial condition to $x(0) = (0.3, 0)$, and tighten the constraints to $[x_{\min}, x_{\max}] = [-0.7, 0.7]$, $[u_{\min}, u_{\max}] = [-7, 7]$. We also adjust the tuning parameters to $Q = \text{diag}(50, 1)$ and $R = 0.01$. With the increased penalization on $x_1$, we expect the control action to quickly compensate to drive $x_1$ to zero, yielding an initially higher magnitude for $x_2 = \dot{x}_1$. However, with the tightened constraint on $x$, we see that LQR fails to satisfy the constraint $x_2 \geq -0.7$ at low time steps, whereas MPC and LBMPC not only satisfy this constraint, but leave a margin for error. This behavior is visually clear by the "saturation" of $x_2$ as seen in the MPC and LBMPC traces in Figure 7. This margin for error comes from the robustified state constraint: even in the worst-case scenario that the disturbance term $g(x(t), u(t))$ is on the edge of the polytope $\mathcal{W}$, the constraint $x_2 \geq -0.7$ will still be satisfied. The robust margin is visibly nonzero due to the fact that the disturbance term is not realizing its worst-case values. Similarly, we see in Figure 8 that the LQR input does not satisfy the constraint $u \geq -7$ (see time $t = 0$), but that MPC and LBMPC do satisfy (and saturate) the robust input constraints.

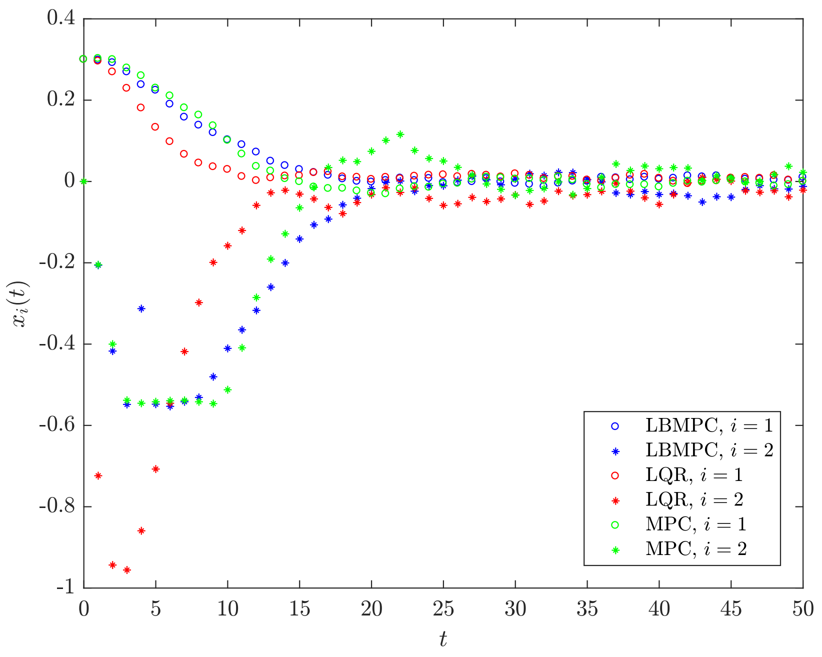

Figure 7: Saturation of the system response. MPC and LBMPC satisfy the state constraint $x_2 \geq -0.7$ (with a robust margin), whereas LQR does not.

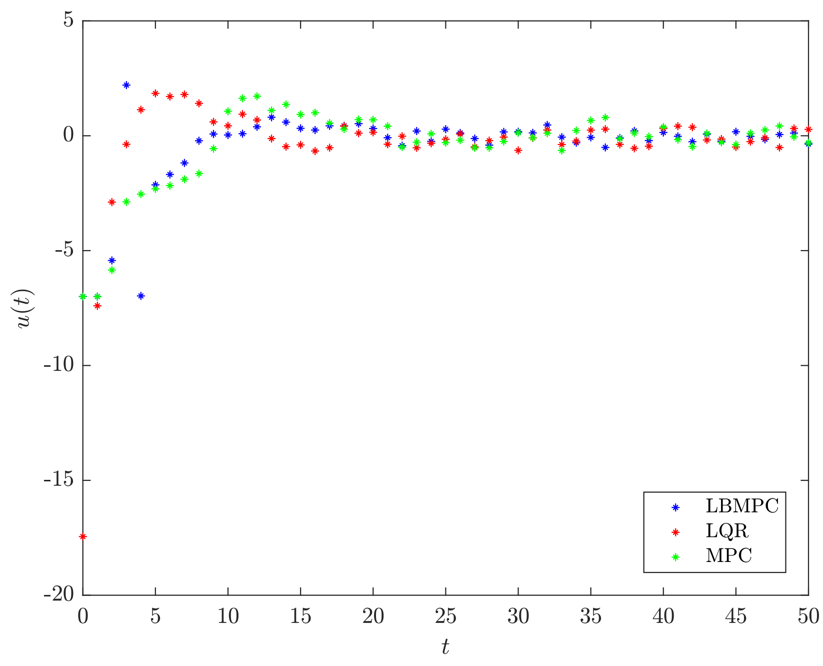

Figure 8: Saturation of the control input. MPC and LBMPC satisfy the input constraint $u \geq -7$, whereas LQR does not (note the LQR input value $u(0) \approx -17.5$).

### 3.2.5 Experiment #3: Learning Advantage

For this experiment, we set $T = 0.05$, $Q = I_n = I_2$, $R = I_m = 1$, $x(0) = (0.3, 0)$, and set the random noise level to zero, i.e. $d_{\min} = d_{\max} = 0$. We also set $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{U} = \mathbb{R}$, i.e. we effectively remove the state and input constraints. We also introduce a linear perturbation to the ground truth system:

$$x(t+1) = f(x(t), u(t)) + d(t) + Hx(t) + Gu(t),$$

where

$$H = F + \begin{bmatrix} 0 & 0 \\ \frac{gT}{L} & 0 \end{bmatrix}.$$

This may be equivalently rewritten as

$$x(t+1) = (A+F)x(t) + (B+G)u(t) + d(t) + \begin{bmatrix} 0 \\ \frac{gT}{L}\sin(x_1(t)) \end{bmatrix},$$

which highlights the fact that $F$ and $G$ may be thought of as perturbations to $A$ and $B$, and that the system maintains its nonlinearity with respect to $x_1$ in the dynamics governing $x_2$. This might very well represent a system whose linearized model has some unknown, constant offset in the matrices $A$ and $B$. We take these offsets to be $F = 0.3A$ and $G = 0.3B$, i.e. the "true" $A$ and $B$ are 30% higher (elementwise) than the $A$ and $B$ in the LTI model used to design the control input $u$. For this experiment, we are not particularly interested in satisfying recursive feasibility and constraint satisfaction in MPC and LBMPC, but rather we are interested in determining whether LBMPC can accurately learn the true perturbed system dynamics and accordingly stabilize the system, even in the case LQR and MPC cannot.

As seen in Figure 9, we find that the answer to the above inquisition is affirmative. That is, we find that LBMPC stabilizes the system when LQR and MPC without learning cannot. At the final time step, the oracle estimation yielded the following approximations for $H$ (and therefore $F$) and $G$:

$$\hat{H} = \begin{bmatrix} 0.3000 & 0.0300 \\ 1.2915 & 0.3109 \end{bmatrix} \implies \hat{F} = \begin{bmatrix} 0.3000 & 0.0300 \\ 0.2915 & 0.3109 \end{bmatrix},$$
$$\hat{G} = \begin{bmatrix} 0.0000 \\ 0.0311 \end{bmatrix},$$

which are seen to very closely match the true values of

$$F = \begin{bmatrix} 0.3 & 0.03 \\ 0.3 & 0.2997 \end{bmatrix},$$
$$G = \begin{bmatrix} 0 \\ 0.03 \end{bmatrix}.$$

These results evidence the fact that the oracle converges to the true modeling error; $\mathcal{O}_t(x(t), u(t)) = H_t x(t) + G_t u(t) \to Hx(t) + Gu(t) \approx Fx(t) + Gu(t) + (0, \frac{gT}{L}\sin(x_1(t))) = g(x(t), u(t))$, where the approximation approaches equality as $x_1(t) \to 0$. This ultimately salvages the stability of the closed-loop system, as depicted in Figure 9. Without this adaptive model of the system, we see that the control system fails to stabilize.

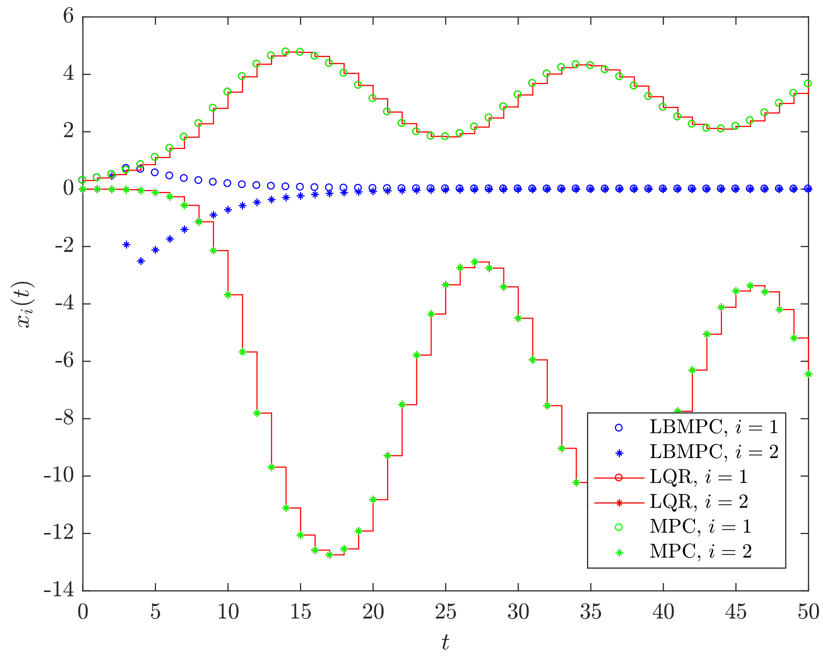

Figure 9: Stabilization with modeling errors. As seen, the oracle in LBMPC is able to adapt the model to match the true system and yield stability. Without learning, LQR and linear MPC give rise to unstable closed-loop systems.

It is important to mention that the use of a linear oracle worked well to learn the dynamics in the case of linear perturbations to the ground truth system. However, in general the modeling error may have strongly nonlinear components, and therefore a linear oracle may fail to adequately learn and model this behavior. Using an oracle which performs better when modeling nonlinear phenomena, such as a neural network, might be warranted in this case.

## 3.3 Inverted Pendulum on a Cart

We now consider the inverted pendulum on a cart, as shown in Figure 10. In this case, the input to the system is a force applied to the body of the cart, denoted $u \in \mathbb{R}$. We denote the cart's lateral position as $x_1 \in \mathbb{R}$, and the pendulum's angular position (from the upright) as $x_3 \in \mathbb{R}$.

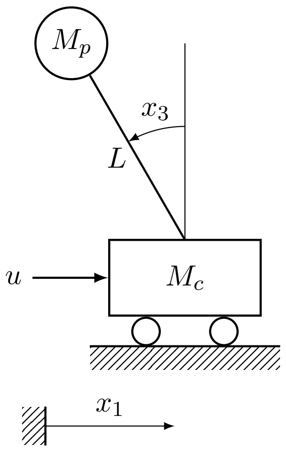

Figure 10: Inverted pendulum on a cart diagram.

### 3.3.1 Dynamic System Model

The position and velocity of the pendulum are given by

$$\begin{aligned} r_p &= (x_1 - L\sin(x_3))e_1 + L\cos(x_3)e_2, \\ \dot{r}_p &= (\dot{x}_1 - L\dot{x}_3\cos(x_3))e_1 - L\dot{x}_3\sin(x_3)e_2, \end{aligned}$$

respectively, where $e_1 = (1,0) \in \mathbb{R}^2$ and $e_2 = (0,1) \in \mathbb{R}^2$. Therefore, the kinetic and potential energy of the system are

$$\begin{aligned} \mathcal{T} &= \frac{1}{2}M_c\dot{x}_1^2 + \frac{1}{2}M_p\|\dot{r}_p\|_2^2 \\ &= \frac{1}{2}M_c\dot{x}_1^2 + \frac{1}{2}M_p(\dot{x}_1^2 - 2L\dot{x}_1\dot{x}_3\cos(x_3) + L^2\dot{x}_3^2), \\ U &= M_pgL\cos(x_3), \end{aligned}$$

respectively, where $M_c$ is the cart mass and $M_p$ is the pendulum mass. Forming the Lagrangian gives

$$\begin{aligned} L &= \mathcal{T} - U \\ &= \frac{1}{2}(M_c + M_p)\dot{x}_1^2 - M_pL\dot{x}_1\dot{x}_3\cos(x_3) + \frac{1}{2}M_pL^2\dot{x}_3^2 - M_pgL\cos(x_3). \end{aligned}$$

Now, the Euler-Lagrange equations for the $x_1$ and $x_3$ coordinates are $\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_1} - \frac{\partial L}{\partial x_1} = u$ and $\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_3} - \frac{\partial L}{\partial x_3} = 0$, where we allow $t \in \mathbb{R}$ to represent continuous time for the current line analysis. These expressions yield the following equations of motion:

$$\begin{aligned} (M_c + M_p)\ddot{x}_1 - M_pL\ddot{x}_3\cos(x_3) + M_pL\dot{x}_3^2\sin(x_3) &= u, \\ L\ddot{x}_3 - \ddot{x}_1\cos(x_3) - g\sin(x_3) &= 0. \end{aligned}$$

Defining $x_2 = \dot{x}_1$ and $x_4 = \dot{x}_3$, we find (after straightforward algebraic manipulation) that the system of equations becomes

$$\begin{aligned}\dot{x} &= \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{bmatrix}\\ &= f_{\text{ct}}(x,u) := \begin{bmatrix} x_2 \\ \frac{1}{M_c+M_p\sin^2(x_3)}(M_pg\cos(x_3)\sin(x_3) - M_pLx_4^2\sin(x_3)+u) \\ x_4 \\ \frac{1}{L(M_c+M_p\sin^2(x_3))}((M_c+M_p)g\sin(x_3) - M_pLx_4^2\cos(x_3)\sin(x_3)+u\cos(x_3))\end{bmatrix}.\end{aligned}$$

Now, denoting $t \in \mathbb{Z}_+$ as the discrete-time index, we can discretize the system as $x(t+1) \approx Tf_{\text{ct}}(x(t),u(t))+x(t)$, where $T$ is the sampling time. This results in the following nonlinear discrete-time system, which we take to be our ground truth:

$$x(t+1) = \begin{bmatrix} x_1(t)+Tx_2(t) \\ x_2(t) + \frac{T}{M_c+M_p\sin^2(x_3(t))}(M_pg\cos(x_3(t))\sin(x_3(t)) - M_pLx_4^2(t)\sin(x_3(t))+u(t)) \\ x_3(t)+Tx_4(t) \\ x_4(t) + \frac{T}{L(M_c+M_p\sin^2(x_3(t)))}((M_c+M_p)g\sin(x_3(t)) - M_pLx_4^2(t)\cos(x_3(t))\sin(x_3(t))+u\cos(x_3(t)))\end{bmatrix}. \tag{8}$$

As described in Section 2, we apply Jacobian linearization to this system about the equilibrium $(x_e,u_e) = (0,0)$ to obtain

$$x(t+1) \approx Ax(t)+Bu(t) = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & \frac{M_pgT}{M_c} & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & \frac{(M_c+M_p)gT}{M_cL} & 1\end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{T}{M_c} \\ 0 \\ \frac{T}{M_cL}\end{bmatrix} u(t). \tag{9}$$

### 3.3.2 Experiment #4: Stabilization

We use the following setup: $T = 0.1$, $M_p = 0.5$, $M_c = 1$, $L = 5$, $g = 10$, $N = 5$, $Q = I_n = I_4$, and $R = I_m = 1$. We leave the problem unconstrained and without additive random noise. Furthermore, we initialize the system at $x(0) = (0, 0, 0.5, 0)$. Without control, we see that the system is unstable, as shown in Figure 11. By implementing LQR and LBMPC, we are able to stabilize the system in the upright position, $(x_e,u_e) = (0,0)$. [Note that this includes driving the cart position and velocity to zero, as well as the pendulum position and velocity. The plots of these trajectories are omitted for brevity.] We see that LBMPC is able to stabilize the pendulum into the upright position faster than LQR.

It is worthwhile to mention that we attempted to perform a "swing up" of the inverted pendulum on a cart by setting $x(0) = (0, 0, \pi, 0)$, i.e. starting the pendulum in the downward position. It was our hope that, as the pendulum swung up, the linear oracle could adequately approximate the changes in the $A$ and $B$ matrices that one would find by linearizing the dynamics iteratively along the path traced out by the pendulum. However, we found that the pendulum swing up would not stabilize in the upright position using LBMPC with a linear oracle (at least with the variety of tuning parameters we tried using). Based on these trials, we conclude that in order to use LBMPC to perform a swing up, the following two more sophisticated approaches might turn out to be fruitful: (i) use an oracle which can better approximate the nonlinearity of the system as the pendulum swings up through its nonlinear region (for example, a neural network may work better), and (ii) update the LQR parameters $K$ and $P$ at each iteration of the LBMPC using the most recent system dynamics approximated via the oracle. In the case that one of these approaches is found to work, a single control scheme can be used to perform the swing up rather than the classical approach of separating the problem into two control design sub-problems: one controller for the swing up and one controller for the stabilization. Although this problem provides a fascinating application of LBMPC, the completion of the task has extended beyond the scope of this report, and therefore we leave it as a suggestion for future work.

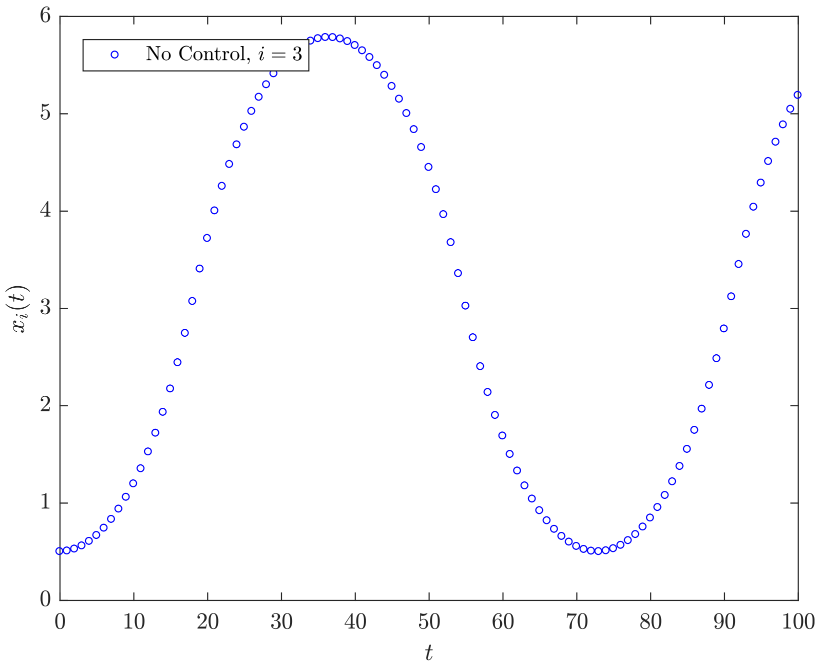

Figure 11: Inverted pendulum on a cart without control input. The trajectory shown depicts the pendulum angle across time and the fact that it oscillates about the downward equilibrium point $x_3 = \pi$ without any control input.

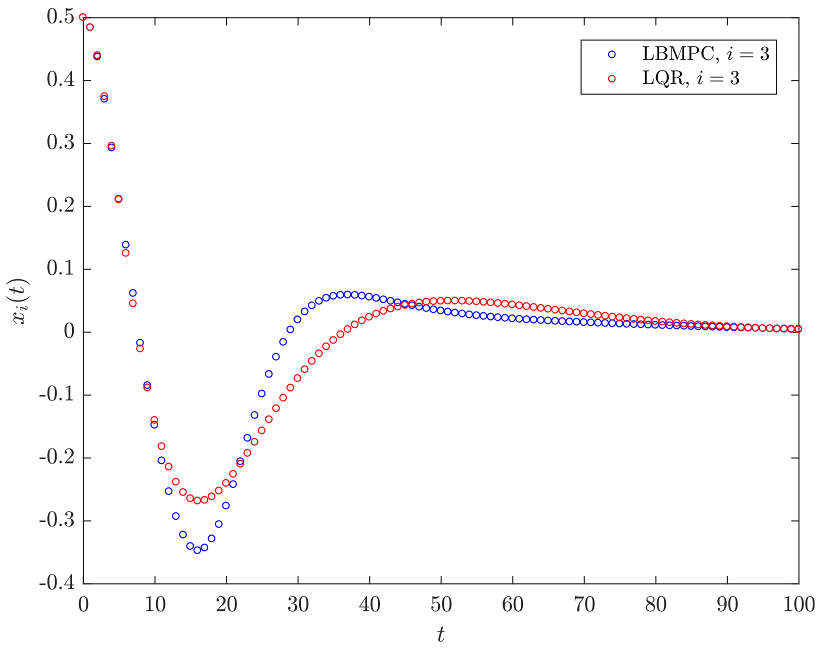

Figure 12: Inverted pendulum on a cart with control input. LBMPC is seen to stabilize the system with a faster settling time than that of LQR.

## 4 Conclusion

In this report, we study the incorporation of machine learning into control theory via learning-based model predictive control (LBMPC). In order of increasing mathematical and practical complexity, we give a theoretical overview of classical linear quadratic regulation (LQR), modern (robust) model predictive control (MPC), and its extension LBMPC. While summarizing each control strategy, we highlight the important theoretical guarantees and performance improvements that each method brings with its increased complexity over the others. These properties are then verified through numerical simulations in which we control two types of inverted pendula to stabilize in their upright position. We find that, as the theory predicts, MPC and LBMPC provide constraint satisfaction not inherently taken into account by LQR. Furthermore, we demonstrate that in the presence of model uncertainties, LQR and MPC can give rise to unstable closed-loop systems, whereas the learning aspect of LBMPC safely adapts the system's internal model to account for the true dynamics and ultimately salvages the stability of the closed-loop system. Ultimately, our experimentation provides empirical evidence to support the use of LBMPC in situations where engineering constraints must be satisfied by a system whose dynamics are not precisely known or exhibit nonlinearities which render previously popular control techniques inadequate.

## References

[1] A. Aswani. *UC Berkeley IEOR 265, Lecture 14: (Robust) linear tube MPC*. URL: `http://courses.ieor.berkeley.edu/ieor265/lecture_notes/ieor265_lec14.pdf`. 2019.

[2] A. Aswani. *UC Berkeley IEOR 265, Lecture 15: Learning-based MPC*. URL: `http://courses.ieor.berkeley.edu/ieor265/lecture_notes/ieor265_lec15.pdf`. 2019.

[3] A. Aswani. *UC Berkeley IEOR 265, Lecture 16: The oracle*. URL: `http://courses.ieor.berkeley.edu/ieor265/lecture_notes/ieor265_lec16.pdf`. 2019.

[4] A. Aswani et al. "Provably safe and robust learning-based model predictive control". In: *Automatica* 49.5 (2013), pp. 1216–1226.

[5] F. Borrelli, C. Jones, and M. Morari. *UC Berkeley ME C231A, Lecture 1: Model predictive control, Part 1 — Introduction*. 2018.

[6] S. Boyd. *Stanford EE363, Lecture 3: Infinite horizon linear quadratic regulator*. URL: `https://web.stanford.edu/class/ee363/lectures/dlqr-ss.pdf`. 2008.