# Reactive Planar Nonprehensile Manipulation with Hybrid Model Predictive Control

**Francois R. Hogan and Alberto Rodriguez**

## Abstract

This paper presents an offline solution and online approximation to the hybrid control problem of planar nonprehensile manipulation. Hybridness and underactuation are key characteristics of this task that complicate the design of feedback controllers. We show that a model predictive control approach used in tandem with integer programming offers a powerful solution to capture the dynamic constraints associated with the friction cone as well as the hybrid nature of contact. We introduce the Model Predictive Controller with Learned Mode Scheduling (MPC-LMS), that leverages integer programming and machine learning techniques to effectively deal with the combinatorial complexity associated with determining sequence of contact modes. We validate the controller design through a numerical simulation study and with experiments on a planar manipulation setup using an industrial ABB IRB 120 robotic arm. Results show that the proposed algorithm achieves closed-loop tracking of a nominal trajectory by reasoning in real-time across multiple contact modalities.

## 1 Introduction

Humans manipulate objects within their hands with impressive agility and ease. While doing so, they also make frequent mistakes from which they recover seamlessly, effectively blurring the line between reactive control and other error-correcting mechanisms. The mechanical complexity of the human hand along with its array of sensors undoubtedly play an important role. However, despite recent advances in the design of complex robotic hands and sensory equipment (tactile sensors, vision markers, proximity sensors, etc.), autonomous robotic manipulation remains far behind human manipulation capability.

We argue that this gap in performance can largely be attributed to robots' inability to use sensor information for real-time control purposes. Whereas humans effectively process and react to information from tactile and vision sensing, robot manipulators are most often programmed in an open-loop fashion, incapable of adapting or correcting their motion. With the recent development of sensing equipment, the question remains: how should robots use sensed information?

In this paper, we address closed-loop control of a nonprehensile manipulation task: planar pushing. Planar pushing is a canonical manipulation skill that incorporates many of the key control challenges typical of robotic manipulation of which we highlight:

1. **Hybridness**. When the pusher interacts with the object, different contact modalities can occur between both entities (e.g. separation, sticking, sliding up, and sliding down). Transitions between these modes result in discontinuities in the dynamics, which complicate controller design.

2. **Underactuation**. Contact interactions can only transmit a limited set of forces. These constraints on the control inputs lead to a dynamical system where reasoning about the instantaneous velocity/acceleration of the pusher is not sufficient to produce an arbitrary velocity/acceleration of the sliding object.

The goal of this paper is to develop a control framework that can handle these constraints under the assumption of full state feedback. Specifically, we propose a real-time model predictive controller that reasons across a sequence of contact modes.

The contributions of this paper are:

- An offline optimal control solution to the hybrid control problem of multiple point planar manipulation as a Mixed-Integer Program.
- An online approximation to the optimal solution based on learning a map from object states to predicted future mode transitions.
- A benchmark comparison of our controller against other hybrid control approaches.
- An experimental demonstration of the controller on a planar manipulation setup.
- An experimental analysis of the performance of the online approximate solution to the hybrid control problem as a function of the controller frequency, the

[1]Department of Mechanical Engineering — Massachusetts Institute of Technology
<fhogan,albertor>@mit.edu

**Corresponding author:**
Francois R. Hogan, MIT 77 Massachussetts Avenue, Cambridge, MA, 02139, USA.

tracking velocity, the planning horizon, the error in coefficient of friction, and the radius of curvature of the track.
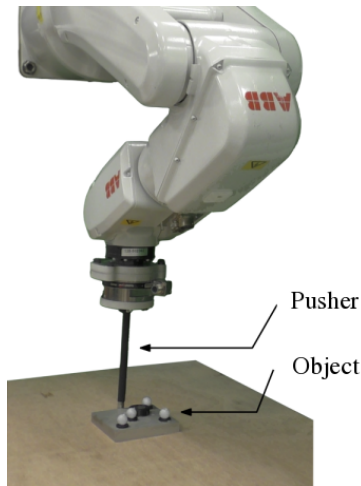


**Figure 1.** Planar manipulation setup. The goal is to control the motion of the object on a flat surface using a velocity controller robotic pusher. The pose of the object is tracked using a Vicon camera system.

Previous robotics research relevant to our work on reactive planar manipulation falls into three broad categories: planar pushing, contact-constrained motion planning, and control. The mathematical development of algorithms that can effectively handle the discontinuities associated with frictional contact interactions is a thread common to these areas. We now review some of the earlier work from these fields.

## 2    Related Work

### 2.1    Planar Pushing

The mechanics of planar pushing manipulation tasks were first developed by Mason (1986). Goyal et al. (1991) introduced the concept of the *limit surface*, a useful geometric representation that describes the contact wrench that can be supported by contact interactions. Close to our work is that of Zhou et al. (2017) that develops a Linear Complementarity Formulation that describes the planar motion of objects subject to robotic pushes.

Lynch and Mason (1996) and Zhou and Mason (2017) introduce motion planning algorithms to find open-loop robotic trajectories achieving a target object pose. Key to the success of these executions are the assumptions that sticking interactions are maintained for the entirety of the push and that the object remains unperturbed during the execution. Dogar and Srinivasa (2011), Dogar and Srinivasa (2012), and Koval et al. (2016) present planning frameworks to grasp objects in cluttered environment that leverage pushing actions to address uncertainty in the pose of objects. Lynch et al. (1992) implements a tactile-based feedback controller for a point pusher-object system to maintain the heading of an object. This PD based controller can stably control the orientation of a pushed object but cannot control its positioning.

### 2.2    Contact-Constrainted Motion Planning

There are ongoing efforts to develop motion planning frameworks that can effectively handle the complexity associated with frictional contact interactions. Over the past decade, this topic has been a central area of focus of the robotics locomotion and manipulation communities.

In particular, in the robotic locomotion community, a common approach consists in formulating the search for gaits as a nonlinear optimization program. Pardo et al. (2017) determines the gait trajectories and impact times under a prespecified contact sequence. Schultz and Mombaur (2009) and Valenzuela (2016) autonomously compute the gait contact sequences are using mixed-integer nonlinear programming, where integer variables are used to encode the contact modes active during the trajectory. Posa et al. (2014) employs a Linear Complementarity Problem formulation to encode the hyrid nature of contact interactions by including contact forces as decision variables within the program. This method has been shown to be effective for path planning of high degree of freedom systems undergoing contact rich interactions.

In the robotic manipulation community, Chavan-Dafle et al. (2014) makes use of sampling-based algorithms to plan robot motions for in-hand manipulation tasks that exploit both sticking and sliding interactions with the environment. This approach relies on open-loop stable executions and rely on an accurate description of contact interactions. Woodruff and Lynch (2017) and Hou et al. (2018) present a graph search algorithm to plan through a sequence of manipulation primitives describing different contact states to achieve a manipulation task. More recently, Toussaint et al. (2018) formulates a task and motion planning framework that can handle complex interactions by formulating the search for contact sequences as a nonlinear mixed-integer optimization program.

Despite the variety of motion planning frameworks available for multi-contact dynamic interactions, these approaches have large computational requirements associated with solving nonlinear and non-convex optimization programs that make them unsuitable for online replanning. A key challenge that remains open in the community is to find control strategies that can replan mode sequences at real-time rates.

### 2.3    Control

A common strategy to deal with hybridness has been to design feedback controllers that rely on a fixed mode schedule set to follow a nominal plan computed offline. Woodruff and Lynch (2017) and Kuindersma et al. (2016) use a linear-quadratic-regulator (LQR) control architecture to stabilize the nominal trajectory subject to a mode sequence searched offline. Pardo et al. (2017) employs a feedback linearization approach to track the planned trajectory. Both of these approaches assume that the mode sequence remains unchanged during the execution of the task. Posa et al. (2016) formulates a constrained LQR approach as a convex optimization program integrating control input constraints. While this approach can handle minor variations in the timing of impacts with the ground, it does not have the ability to alter its planned mode sequences over a finite
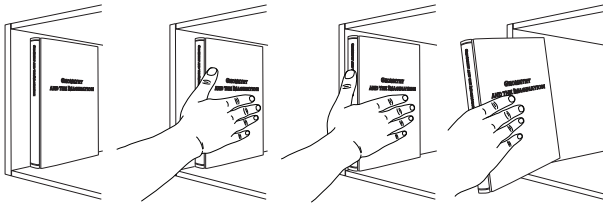
**Figure 2.** Depiction of hybridness. Animation of a simple manipulation task that exploits multiple contact modes. First, the hand sticks to the book and drags it backwards exploiting friction. Second, thumb and fingers slide to perform a regrasp maneuver. Finally, the book is retrieved from the shelf using a stable grasp.



**Figure 3.** Depiction of underactuation. Where we interact with an object through contact, we can only transfer a limited set of forces. When pushing a coffee mug with a finger, the finger can only push on the object and cannot pull. Underactuation constrains the possible motions of the cup that can be impressed by the finger.

horizon, for example to change its gait sequence or footstep plan. It is important to note that while this philosophy might be relatively sensible in locomotion (a gait precisely defines a sequence of ordering of contacts) it is very limiting in manipulation, where a significant part of the richness from reactive behavior comes from quickly adapting to unexpected contact events.

An important drawback of the aforementioned approaches is the controller's inability to replan the mode sequences in real-time. There have been recent efforts towards designing hybrid feedback control architectures that can reason accross system discontinuities (Buehler et al. 1994; Murphey and Burdick 2004). Of particular interest to this research are Bemporad and Morari (1999) and Lazar et al. (2006) that formulate the hybrid MPC problem as a mixed-integer program that integrates both continuous and discrete variables. The controllers developed in these works establish closed-loop stability by reasoning across multiple contact modes, however struggle to achieve real-time rates even for small dimensional systems as the scalability of the approach is limited by the number of hybrid states and the length of the control horizon. Another approach that shows promise is explicit MPC, a multiparametric programming techniques that computes the optimal control action offline as an "explicit" function of the state and reference vectors, so that on-line operations reduce to a function evaluation (Bemporad et al. 2002; Alessio and Bemporad 2006; Oberdieck and Pistikopoulos 2015). While these approaches enable real-time control in theory, in practice they are associated with large offline computational requirements that scale poorly with the dimensionality of the system, the number of hybrid modes and the length of control horizon. These approaches require enumerating the complete set of feasibility switching sequences offline through a backwards reachability analysis.

# 3 CHALLENGES OF CONTROL THROUGH CONTACT

This work aims to design a closed-loop controller that can reason about the frictional contact interactions arising between a robot's end-effector and a manipulated object. Systems undergoing frictional contact with their environment present two key challenges for controller design: hybridness and underactuation.
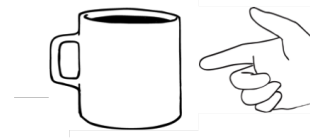
## 3.1 Hybridness

When in contact, object and manipulator interact in different contact modes. For example, during manipulation tasks, the object can slip within the fingers of the gripper, the gripper can throw the object in the air, or the gripper can perform pick and place maneuvers. These manipulation actions correspond to different contact interaction modes, characterized by the sliding, sticking, or separation of the individual contacts. The hybridness associated with the transitions between modes can result in a non-smooth dynamical system. This complicates the design of feedback controllers since the vast majority of standard control techniques rely on smoothness of the dynamical model.

In many applications involving hybrid dynamical systems, this difficulty is overcome by setting a schedule of mode transitions of the controller offline. This limitation prevents the controller to fully exploit the dynamics of the system in response to external perturbations. Furthermore, for robotic manipulation tasks, the mode scheduling is often not known a priori and can be challenging to predict. In such cases, we must rely on the controller to decide, during execution, what interaction mode is most beneficial to the task. Figure 2 illustrates the example of picking a book from a shelf. The hand interacts with the book in a complex manner. It is difficult to say when fingers and palm stick or slide, but those transitions not only happen, but are necessary to pick the book. Likely the hand initially sticks to the book and drags it backwards exploiting friction. Then, the thumb and fingers swiftly slide to regrasp the book. Finally, the book is retrieved from the shelf using a stable grasp. For such manipulation tasks where the motion is not periodic, determining a fixed mode sequencing strategy is not obvious and often impractical. Mistakes committed during execution or encountered external perturbations will surely require that the mode sequencing be altered.

## 3.2 Underactuation

Underactuation is due to the fact that contact interactions can only transmit a limited set of forces and torques to the object. As such, the controller must choose only among the forces that can physically be realized. For example, the normal forces commanded should be positive, as contact interactions can only "push" and cannot "pull." In order to achieve this, it is required to explicitly impose the physical constraints associated with contact interactions in the controller design. The principles and conditions that must be considered include Coulomb's frictional law, the non-penetrating condition, and the principal of maximum

dissipation. These concepts are described in Stewart and Trinkle (1996) and further detailed in Section 5.5. The consequence of limited control authority is that the controller must reason beyond instantaneous actuation by considering the long term consequences of control actions.

## 4   NOMENCLATURE

We describe here the notation used in the paper:

- $\mathcal{F}_a$: Inertial reference frame fixed to the ground.
- $\mathcal{F}_b$: Body reference frame fixed to the object.
- $C$: Number of contact points (indexed by $c$).
- $\mathbf{w} = [f_x \quad f_y \quad \tau]^\mathsf{T}$: Applied wrench on the object resolved in the body frame.
- $\mathbf{t} = [v_x \quad v_y \quad \omega]^\mathsf{T}$: Object twist resolved in the body frame.
- $\mathbf{J}_c$: Jacobian matrix associated with the contact point $c$ resolved in the body frame.
- $\mathbf{N} = [\mathbf{J}_1^\mathsf{T}\mathbf{n}_1 \quad \ldots \quad \mathbf{J}_C^\mathsf{T}\mathbf{n}_C]^\mathsf{T}$: Matrix of object normal vectors at contact points resolved in body frame.
- $\mathbf{T} = [\mathbf{J}_1^\mathsf{T}\mathbf{t}_1 \quad \ldots \quad \mathbf{J}_C^\mathsf{T}\mathbf{t}_C]^\mathsf{T}$: Matrix of object tangent vectors at contact points resolved in body frame.
- $\mathbf{f}_n = [f_{n,1} \quad \ldots \quad f_{n,C}]^\mathsf{T}$: Vector of applied normal force magnitudes at contact points resolved in body frame.
- $\mathbf{f}_t = [f_{t,1} \quad \ldots \quad f_{t,C}]^\mathsf{T}$: Vector of applied tangential force magnitudes at contact points resolved in body frame.
- $\boldsymbol{\phi} = [\phi_1 \quad \ldots \quad \phi_C]^\mathsf{T}$: Vector of relative angles of pusher relative to body frame.
- $\mathbf{x} = [x \quad y \quad \theta \quad \boldsymbol{\phi}^\mathsf{T}]^\mathsf{T}$: System state vector: position and orientation of the object in $\mathcal{F}_a$ as well as the relative angles of the contact points relative to $\mathcal{F}_b$.
- $\mathbf{u}_f = [\mathbf{f}_n^\mathsf{T} \quad \mathbf{f}_t^\mathsf{T}]^\mathsf{T}$: Vector of commanded contact forces.
- $\mathbf{u}_\phi = \dot{\boldsymbol{\phi}}$: Vector of commanded angular velocities resolved in body frame.
- $\mathbf{u} = [\mathbf{u}_f^\mathsf{T} \quad \mathbf{u}_\phi^\mathsf{T}]^\mathsf{T}$: Control input.

Note that we choose to parameterize the location of the pusher on the surface of the object by the angle $\phi$. This assumes a bijective mapping between the angle $\phi$ and the surface of the object, which is not always true. Whereas an arc-length parametrization would be more general, the parametrization with the angle $\phi$ works for all "start-shaped" objects and simplifies the description of the kinematics of pushing.

## 5   PLANAR MANIPULATION

This paper studies planar manipulation, a nonprehensile task where the goal is to control the motion of a sliding object through frictional contact interactions. Planar manipulation is an interesting dynamical system to study controller design since the source of actuation arises purely from friction. Moreover, this system highlights the importance of reasoning in real-time across contact modes. By perturbing the system and altering the contact state, the success of the task depends on the ability of the controller to modify its originally planned contact state in an online fashion.

This section describes the mechanics of planar manipulation and derives the motion equations used in Section 6
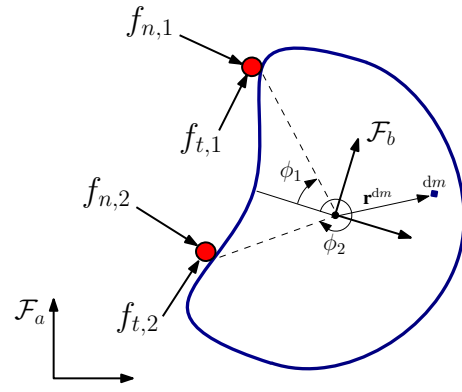


**Figure 4.** Free body diagram of a sliding object with $C = 2$ contact points.

for controller design. This model generalizes to an arbitrary number of contact points and arbitrary object shapes.

### 5.1   Kinematics

Consider the pusher-object system in Fig. 4. The pose of the object is given by

$$\mathbf{p} = \left[ \begin{array}{ccc} x & y & \theta \end{array} \right]^\mathsf{T},$$

where $x$ and $y$ denote the cartesian coordinates of the center of mass of the object and $\theta$ its orientation relative to the inertial reference frame $\mathcal{F}_a$. Assuming the point $c \in \{1, \ldots, n\}$ associated with the pusher remains in contact with the object at all time, the position of the contact point $c$ relative to the object resolved in the body frame $\mathcal{F}_b$ is

$$\mathbf{r}_c = \left[ \begin{array}{cc} x_c & y_c \end{array} \right]^\mathsf{T}.$$

For shapes that can be parametrized radially, the position of the contact point can be described in terms of the radial distance $r = f(\xi)$

$$\mathbf{r}_c = \left[ \begin{array}{cc} -f(\xi)\cos\phi_c & f(\xi)\sin\phi_c \end{array} \right]^\mathsf{T},$$

where the angle $\phi_c$ describes the location of the contact point along the perimeter and $\xi$ is used to parametrize the shape of the object radially.

### 5.2   Quasi-Static Approximation

Applying Newton's second law in the $x - y$ plane of $\mathcal{F}_a$ yields the motion equations

$$\mathbf{H}\ddot{\mathbf{p}} = \mathbf{f}_G + \mathbf{w}, \tag{1}$$

where $\mathbf{H}$ is the inertia matrix of the system, $\mathbf{w}$ is the generalized frictional force applied by all pushers on the object, and $\mathbf{f}_G$ is the generalized frictional force applied by the ground on the object. The quasi-static assumption observes that at low velocities, frictional contact forces dominate and inertial forces do not have a decisive role in determining the motion of the object Mason (2001). Under this assumption, kinematic and frictional forces are in equilibrium with the applied frictional wrench by the pusher is of equal magnitude and opposite direction to the ground planar frictional force (i.e., $\mathbf{w} = -\mathbf{f}_G$). The quasi-static assumption leads to a simplified analysis of the motion
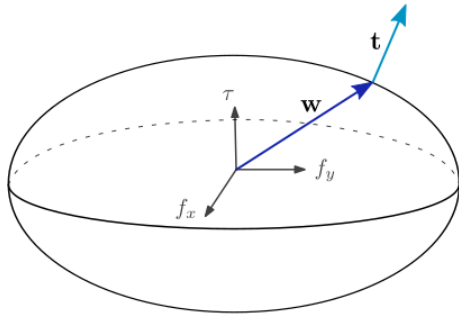
**Figure 6.** Friction cone constraint. The applied force must remain within the blue shaded region.

**Figure 5.** Depiction of the limit surface. The limit surface (ellipsoid in the figure) describes the set of forces and moments that can be transmitted by a patch contact. By the principle of maximal dissipation, the object twist is perpendicular to the limit surface of the applied frictional wrench $\mathbf{w}$.

of the pushed object by using a force balance between the applied forces on the object and the frictional forces between the object and the ground.

Note that the term $\mathbf{H}\ddot{\mathbf{p}}$ could easily be integrated into the control formulation presented in Section 6. The quasi-static assumption however presents advantages as it leads to a direct mapping between the motion of the object and the motion of the pusher and has useful properties, such as invariance of the motion equations to the magnitude of the planar coefficient of friction. This also proved desirable from an experimental implementation standpoint when using a position controlled robotic manipulator.

### 5.3 Limit Surface

The limit surface is a geometric representation that bounds the set of all possible frictional forces and moments that can be sustained by a frictional interface. First introduced in Goyal et al. (1991), under the quasi-static assumption, the limit surface maps the applied frictional force on an object to its resulting velocity. We use a convex quadratic approximation to the limit surface as described in Zhou et al. (2016), where the limit surface can be expressed as the sublevel set

$$H(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{L}\mathbf{w},$$

where $\mathbf{L}$ is positive definite and the applied pusher wrench is denoted by $\mathbf{w}$. In this paper, we use an ellipsoidal approximation to the limit surface by Lee and Cutkosky (1991), where the semi-principal axes are given by $f_{max}$, $f_{max}$, and $m_{max}$ defined by

$$f_{max} = \mu_g mg$$

and

$$m_{max} = \frac{\mu_g mg}{A} \int_{\mathcal{A}} \|\mathbf{r}^{\mathrm{d}m}\| \, \mathrm{d}A.$$

The term $\mu_g$ describes the coefficient of friction between the object and the ground, $m$ is the mass of the object, $g$ is the gravitational acceleration, $A$ is the surface area of the object exposed to friction, and $r^{\mathrm{d}m}$ is the position of an infinitesimal mass $\mathrm{d}m$ relative to the origin of the object. The ellipsoidal approximation captures well the shape of the limit surface for object-surface contact interactions that
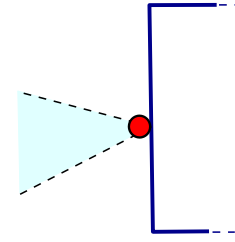
have uniform pressure distributions and yields a convenient invertible analytical form Lee and Cutkosky (1991).

### 5.4 Motion Model

The principle of maximal dissipation states that an object will react to an applied force by moving in the direction that maximizes the system's dissipated power. In practice, this establishes a constraint between the applied force on the object and the resulting velocity of the object. Given the convex limit surface described by $H(\mathbf{w})$, the resulting object twist subjected to an external force will be in the perpendicular direction to the limit surface,

$$\mathbf{t} = \boldsymbol{\nabla} H(\mathbf{w}) = \mathbf{L}\mathbf{w}, \tag{2}$$

where the applied frictional wrench by the set of pushers to the object is

$$\mathbf{w} = \sum_{c=1}^{C} \mathbf{J}_c^\mathsf{T} \left(\mathbf{n}_c f_{n,c} + \mathbf{t}_c f_{t,c}\right) \tag{3}$$

$$= \underbrace{\begin{bmatrix} \mathbf{J}_1^\mathsf{T}\mathbf{n}_1 & \dots & \mathbf{J}_C^\mathsf{T}\mathbf{n}_C \end{bmatrix}}_{\mathbf{N}} \begin{bmatrix} f_{n,1} \\ \vdots \\ f_{n,C} \end{bmatrix} \tag{4}$$

$$+ \underbrace{\begin{bmatrix} \mathbf{J}_1^\mathsf{T}\mathbf{t}_1 & \dots & \mathbf{J}_C^\mathsf{T}\mathbf{t}_C \end{bmatrix}}_{\mathbf{T}} \begin{bmatrix} f_{t,1} \\ \vdots \\ f_{t,C} \end{bmatrix} \tag{5}$$

$$= \mathbf{B}\mathbf{u}_f, \tag{6}$$

with

$$\mathbf{B} = [\mathbf{N}\ \mathbf{T}], \quad \mathbf{J}_c = \begin{bmatrix} 1 & 0 & -y_c \\ 0 & 1 & x_c \end{bmatrix}, \quad \mathbf{u}_f = [\mathbf{f}_n^\mathsf{T}\ \mathbf{f}_t^\mathsf{T}]^\mathsf{T},$$

and where $\mathbf{n}_c$ and $\mathbf{t}_c$ denote the normal and tangential directions of the applied forces in $\mathcal{F}_b$ and $f_{n,c}$ and $f_{t,c}$ represent the magnitudes of the normal and tangential applied forces in $\mathcal{F}_b$.

Consider the planar manipulation system with multiple contact points shown in Fig. 4. The motion equations of the system can be expressed as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

$$= \begin{bmatrix} \mathbf{R}\mathbf{t} \\ \mathbf{u}_\phi \end{bmatrix} = \begin{bmatrix} \mathbf{R}\mathbf{L}\mathbf{B}\mathbf{u}_f \\ \mathbf{u}_\phi \end{bmatrix} = \begin{bmatrix} \mathbf{R}\mathbf{L}\mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{u}, \tag{7}$$

with

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

(a) Sticking. The relative velocity between the pusher and object is zero. (Kinematic constraint).

(b) Sliding left. The frictional force lies on the lower boundary of the friction cone. (Force constraint).

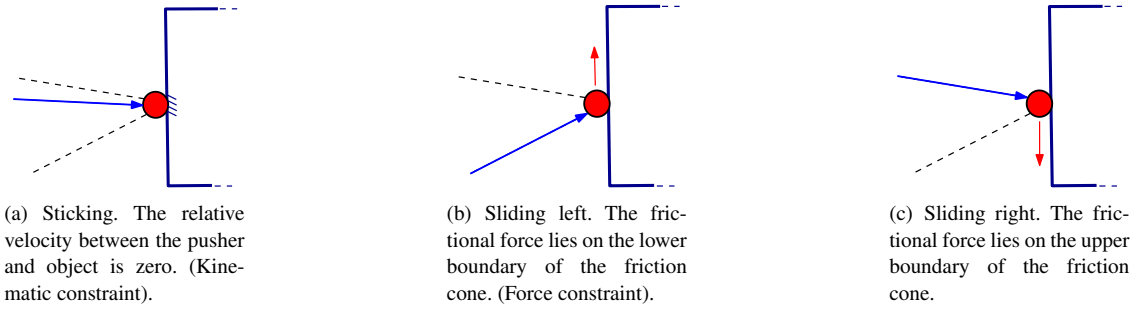(c) Sliding right. The frictional force lies on the upper boundary of the friction cone.

**Figure 7.** Mode dependent constraints following Coulomb's frictional interaction law.

where $\mathbf{R}$ is a rotation matrix and $\mathbf{u}_\phi$ represents the relative sliding velocity between pusher and object. Equation (7) assumes that all points maintain contact with the sliding object and that the applied forces satisfy physical interaction laws as presented in Section 5.5.

## 5.5 Frictional Constraints

The motion equations in Eq. (7) do not enforce that the reaction forces between manipulator and sliding object are feasible. For example. if the input $\mathbf{u}_f = [\mathbf{f}_n^\top \ \mathbf{f}_t^\top]^\top$ is unconstrained, negative normal forces could be applied to the object, which is physically inconsistent, since contact interactions cannot transmit such forces. To ensure that the motion equations are associated with physically reasonable behavior, we must impose constraints on the control input $\mathbf{u}$, ensuring that the motion model obeys contact interactions laws. An important property of contact mechanics is that the physical constraints dictating the magnitude and direction of the frictional forces vary with the contact interaction mode.

**Friction Cone** In accordance with Coulomb's frictional law, the following constraints on the inputs are always satisfied independently of the contact mode:

$$\mathcal{C}_0 : \begin{cases} f_{n,c} \geq 0, \\ |f_{t,c}| \leq \mu_p f_{n,c} \end{cases} \tag{8}$$

implying that each pusher can only exert a compressive force on the object and that the net frictional force applied on the object remains within the bounds of the friction cone in Fig. 6. In addition, we must enforce constraints that depend on the contact interaction mode.

**Sticking** When the pusher is sticking relative to the object, the tangential velocity is stationary, as in Fig. 7(a)

$$\mathcal{C}_1 : \quad \dot{\phi}_c = 0 \tag{9}$$

**Sliding Left** When the pusher is sliding left relative to the object, the tangential velocity is strictly positive and the frictional force must remain on the right hand side of the friction cone, as in Fig. 7(b)

$$\mathcal{C}_2 : \begin{cases} \dot{\phi}_c > 0, \\ f_{t,c} = \mu_p f_{n,c} \end{cases} \tag{10}$$

**Sliding Right** When the pusher is sliding right relative to the object, the tangential velocity is strictly negative and the frictional force is constrained to remain on the left hand side of the friction cone, as in Fig. 7(c)

$$\mathcal{C}_3 : \begin{cases} \dot{\phi}_c < 0, \\ f_{t,c} = -\mu_p f_{n,c}. \end{cases} \tag{11}$$

## 5.6 Pusher Force-Velocity Mapping

The motion equations in Eq. (7) are expressed in terms of the object state $\mathbf{x}$, the applied forces $\mathbf{u}_f$, and the relative pusher sliding velocities $\mathbf{u}_\phi$. This mapping from applied forces to object velocity is desirable from a controller design perspective as the motion equations in Eq. (7) are independent from the contact mode. This is not the case for the relation between pusher velocity and object velocity, as described in  Hogan and Rodriguez (2016).

In practice, it is easier for most position controlled robots to control the robot kinematically (i.e. velocity control) rather than in force control. As such, once the target control $\mathbf{u}$ is computed, it is necessary to map it back to a desired robot velocity $\mathbf{v}_{p,c}$, resolved in the body frame. The robot velocity is linearly related to the object velocity through the kinematic relation

$$\begin{align} \mathbf{v}_{p,c} &= \mathbf{J}_c \mathbf{t} + \dot{\mathbf{r}}_c \tag{12} \\ &= \mathbf{J}_c \mathbf{L} \mathbf{B} \mathbf{u}_f + \frac{\partial \mathbf{r}_c}{\partial \phi} \dot{\phi}_c \tag{13} \\ &= \underbrace{\left[ \mathbf{J}_c \mathbf{L} \mathbf{B} \quad \frac{\partial \mathbf{r}_c}{\partial \phi} \mathbf{1}_c \right]}_{\mathbf{G}_c} \mathbf{u}. \tag{14} \end{align}$$

We can think of $\mathbf{v}_{p,c}$ as the velocity of the contact point on the pusher side, $\mathbf{J}_c \mathbf{t}$ the velocity of the contact point on the object side, and $\dot{\mathbf{r}}_c$ their difference, i.e. the velocity of sliding.

## 5.7 Linearization

This section briefly describes the linearization of the motion equations in Section 5.4 about a given nominal trajectory. We will see that this linearization yields approximate dynamic equations, which can be enforced as linear matrix inequalities in an optimization program and are computationally tractable. This will be essential for the real-time execution of the controller design presented in Section 6. Consider a feasible nominal trajectory $\mathbf{x}^\star(t)$ of the sliding object with nominal control input $\mathbf{u}^\star(t)$ of the pusher. The notation $(\cdot)^\star$ is used to evaluate a term at the equilibrium state and $(\bar{\cdot})$ is used to denote a perturbation about the equilibrium state. The linearization of motion equations Eq. (7) about a nominal trajectory is

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}(t)\bar{\mathbf{x}} + \mathbf{B}(t)\bar{\mathbf{u}},$$

where $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^\star$, $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^\star$, and

$$\mathbf{A}(t) = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}^\star(t), \mathbf{u}^\star(t)}, \quad \mathbf{B}(t) = \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}^\star(t), \mathbf{u}^\star(t)}. \tag{15}$$

The terms $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are computed symbolically using the function `jacobian()` in `matlab`, where $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is given by Eq. (7).

# 6 CONTROLLER DESIGN

In this section, we present a closed-loop controller design that can stabilize the motion of a pushed object about a nominal trajectory. A key feature of this controller is its ability to reason across a sequence of contact modes to fully exploit the dynamics of the system. The proposed controller determines the desired pusher motion (applied force and velocity) at each time step based on the sensed pose of the object.

The following section first presents an offline optimal control solution to the hybrid control problem by formulating a Mixed-Integer Quadratic Program (MPC-MIQP). We then introduce a novel hybrid controller termed Model Predictive Control with Learned Mode Schedules (MPC-LMS), that leverages solutions from the MPC-MIQP program offline to enable real-time control.

## 6.1 Hybrid Model Predictive Control

A successful feedback controller must:

1. Allow for sliding and sticking at contact.
2. Recover from applied perturbations to the nominal trajectory.
3. Be fast enough to solve online.

To satisfy these requirements, we elect to use an MPC formulation, which takes the form of an optimization program over the control inputs during a finite time horizon $t_0, \ldots, t_N$. The decision variables of the optimization program include the perturbed states of the system about the nominal trajectory for $N$ time steps $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N$ and the perturbed control inputs $\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_{N-1}$. The goal is represented by a finite-horizon cost-to-go function that we minimize subject to the constraints on the control inputs and the dynamics of the system detailed in Section 5. We express the cost-to-go for $N$ time steps as:

$$J(\bar{\mathbf{x}}_i, \bar{\mathbf{u}}_i) = \bar{\mathbf{x}}_N^\mathsf{T} \mathbf{Q}_N \bar{\mathbf{x}}_N + \sum_{i=0}^{N-1} \left( \bar{\mathbf{x}}_{i+1}^\mathsf{T} \mathbf{Q} \bar{\mathbf{x}}_{i+1} + \bar{\mathbf{u}}_i^\mathsf{T} \mathbf{R} \bar{\mathbf{u}}_i \right). \tag{16}$$

The terms $\mathbf{Q}$, $\mathbf{Q}_N$, and $\mathbf{R}$ denote weights matrices associated with the error state, final error state, and control inputs. These represent standard objectives in a trajectory optimization problem where the planned trajectory must reach the goal, the intermediate trajectory should approach the nominal trajectory, and the actuation effort should be minimized. We subject the search for optimal control inputs to the constraints:

$$\forall (i, c) \begin{cases} \bar{\mathbf{x}}_{i+1} = \bar{\mathbf{x}}_i + h \left[ \mathbf{A}_i \bar{\mathbf{x}}_i + \mathbf{B}_i \bar{\mathbf{u}}_i \right], \\ f_{n,ci} \geq 0, \\ f_{t,i} \geq 0, \\ \mu f_{n,i} \geq f_{t,i}, \end{cases} \tag{17}$$

developed in Section 5 where $i$ is the time index and $c$ denotes the contact point. The first constraint is the linearization of the dynamic equations of motion, with $\mathbf{A}_i$ and $\mathbf{B}_i$ from Eq. (7). This leads to linear constraints that are computationally tractable for real-time execution.

Additionally, depending on the contact mode at play at each iteration $i$ of the prediction finite horizon, the controller enforces the extra constraints

$$\text{if Mode}(i) = \text{Sticking:} \quad \begin{cases} \dot{\phi}_{ci} = 0, \end{cases} \tag{18}$$

$$\text{if Mode}(i) = \text{Sliding up:} \quad \begin{cases} \dot{\phi}_{ci} > 0, \\ \mu_p f_{n,ci} = f_{n,ci}, \end{cases} \tag{19}$$

$$\text{if Mode}(i) = \text{Sliding down:} \quad \begin{cases} \dot{\phi}_{ci} < 0, \\ \mu_p f_{n,ci} = f_{n,ci}, \end{cases} \tag{20}$$

where the term $\text{Mode}(i)$ denotes the contact mode of interaction at the $i^\text{th}$ step of the prediction horizon.

The constraints in Eqs. (18), (19), and (20) depend on the contact mode and complicate the search for optimal and feasible control inputs. Contact modes and control inputs must be chosen simultaneously. In its naive form, this problem takes the form of a tree of optimization programs with $(Mn)^N$ possible contact schedules, where $M$ denotes the number of possible contact modes for each contact point, $n$ the number of contact points, and $N$ the length of the control horizon. Each branch of the tree requires solving a convex quadratic program, which is too computationally expensive to solve online.

## 6.2 Mixed-Integer Quadratic Program (MPC-MIQP)

The combinatorial hybrid nature of the pusher-object dynamics can be modeled by introducing integer decision variables into the optimization program, as is commonly done in mixed-integer programming. The resulting mixed-integer quadratic program (MIQP) can be solved efficiently using commercial numerical tools, such as Gurobi (Gurobi Optimization (2015)).

In the case of the pusher-object system, we introduce the integer variables: $z_{1i} \in \{0, 1\}$, $z_{2i} \in \{0, 1\}$, and $z_{3i} \in \{0, 1\}$, where $z_{1i} = 1$, $z_{2i} = 1$, or $z_{3i} = 1$ indicate that the
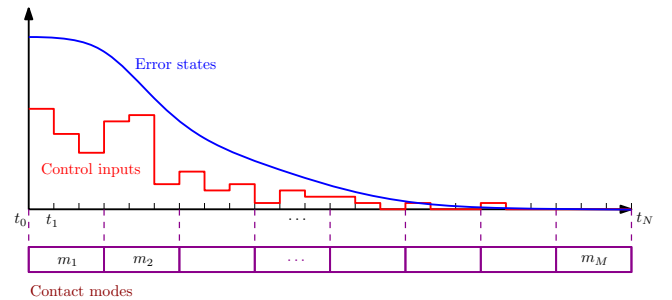


**Figure 8.** Hybrid MPC framework. A sequence of control inputs is computed that will drive the predicted states to the reference trajectory while simultaneously finding the schedule of optimal hybrid mode transitions $\mathbf{m} = \{m_1, \ldots, m_M\}$. The control input $\bar{\mathbf{u}}_0 + \mathbf{u}_0^\star$ is applied to the system.
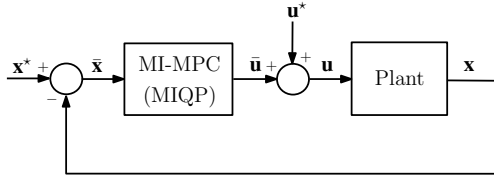
**Figure 9.** Block diagram of the hybrid controller design proposed in Eq. (21). The resulting MPC controller design involves solving a non-convex mixed-integer quadratic program. Note that the integer variables are internal to the controller, necessary to represent the hybrid dynamics, and as an aid to find the optimal control sequence.

contact interaction mode at step $n$ is either sticking, sliding up, or sliding down, respectively. The hybrid MPC problem with additional integer variables denoting the contact modes is detailed as:

**Optimization Problem 1 (MIQP):** Given the current error state $\bar{\mathbf{x}}_0$ and nominal trajectory $(\mathbf{x}_i^\star, \mathbf{u}_i^\star)$, solve

$$\min_{\bar{\mathbf{x}}_i,\ \bar{\mathbf{u}}_i,\ \mathbf{z}_i} \quad \bar{\mathbf{x}}_N^\mathsf{T} \mathbf{Q}_N \bar{\mathbf{x}}_N + \sum_{i=0}^{N-1} \left( \bar{\mathbf{x}}_{i+1}^\mathsf{T} \mathbf{Q} \bar{\mathbf{x}}_{i+1} + \bar{\mathbf{u}}_i^\mathsf{T} \mathbf{R} \bar{\mathbf{u}}_i + \mathbf{z}_i^\mathsf{T} \mathbf{W}_i \mathbf{z}_i \right)$$

$$\text{subject to} \quad \bar{\mathbf{x}}_{i+1} = \bar{\mathbf{x}}_i + h\left[\mathbf{A}_i \bar{\mathbf{x}}_i + \mathbf{B}_i \bar{\mathbf{u}}_i\right],$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_0,$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_1 \quad \text{if } c \text{ is sticking (i.e., } z_{1c,i} = 1),$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_2 \quad \text{if } c \text{ is sliding left (i.e., } z_{2c,i} = 1),$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_3 \quad \text{if } c \text{ is sliding right (i.e., } z_{3c,i} = 1),$$
$$z_{1c,i} + z_{2c,i} + z_{3c,i} = 1,$$

$$(21)$$

with $\mathbf{z}_i = [z_{1c,i}, z_{2c,i}, z_{3c,i}]^\mathsf{T}$. The term $\mathbf{W}_i$ is a weight matrix that can be used to penalize contact switches between the nominal trajectory and the corrected actions. We enforce that the sum of integers values must be unity at each time step to ensure that only one mode can be active at a time.

The resulting mixed-integer optimization program is simultaneously tasked with finding the optimal hybrid mode schedule during the prediction horizon ($\mathbf{z}_i$) and the optimal control sequence ($\mathbf{u}_i$). In practice, we employ the big-M formulation (Nemhauser and Wolsey (1988)) to formulate the problem, where $M$ is a large scalar value used to activate and deactivate the contact mode dependent constraints, through a set of linear equations.

To speed up computation and reduce the number of integer variables, it is useful to constraint adjacent time steps within a prediction horizon to have the same contact mode. This is shown in Fig. 8, where the agglomerated mode sequence
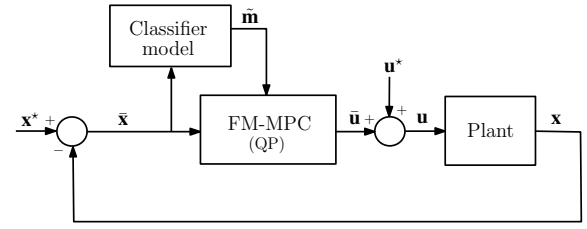
---

**Algorithm 1:** Learned Mode Scheduling (LMS)

**input** : $\bar{\mathbf{x}}(t_0), \mathbf{x}^\star(t), \mathbf{u}^\star(t), \mathbf{\Sigma}$
**output:** $\bar{\mathbf{u}}(t_0)$

**Offline**;
$M_{\boldsymbol{\theta}} = \text{Mode Classifier}(\mathbf{\Sigma})$

**Online**;
$\tilde{\mathbf{m}} = \mathbf{M}_{\boldsymbol{\theta}}(\bar{\mathbf{x}}(t_0))$;
$\bar{\mathbf{x}}_i, \bar{\mathbf{u}}_i, J = \textit{FM-MPC}(\bar{\mathbf{x}}(t_0), \mathbf{x}^\star(t), \mathbf{u}^\star(t), \tilde{\mathbf{m}})$
$\bar{\mathbf{u}}(t_0) = \bar{\mathbf{u}}_0$

---



**Figure 10.** Block diagram of the hybrid controller design with learned mode schedule classifier. The resulting MPC controller design is a convex quadratic program.

$\mathbf{m} = \{m_1, \ldots, m_M\}$ is introduced, with $m_m \in \{S, L, R\}$ denoting sticking, sliding left, and sliding right.

### 6.3 Learned Mode Scheduling (MPC-LMS)

The Mixed-Integer Quadratic Formulation presented in Section 6.2 offers a powerful solution to stabilize hybrid and underactuated dynamical systems and achieve closed-loop tracking of a nominal trajectory. However, its computation requirements remain too great for real-time control (see Section 7 for more detail). This section introduces a controller design that separates the search for the mode schedule from the optimal control sequence. We describe how this splitting of the problem permits us to leverage offline computations to achieve real-time control (see Section 7 for more details).

Consider the controller design architecture proposed in Fig. 10. Suppose that given the state error $\bar{\mathbf{x}}$, we had access to an oracle function that returned an effective mode schedule $\tilde{\mathbf{m}} = \{m_1, m_2, \ldots, m_M\}$ to be enforced during the prediction horizon. In such a case, the control problem would reduce to determining the optimal control inputs under the prescribed mode sequence by solving Optimization Problem 2. Although we do not have direct access to a real-time function that determines the optimal mode schedule, we can query Optimization Problem 1 as much as desired offline to find optimal mode sequences given errors in the input state.
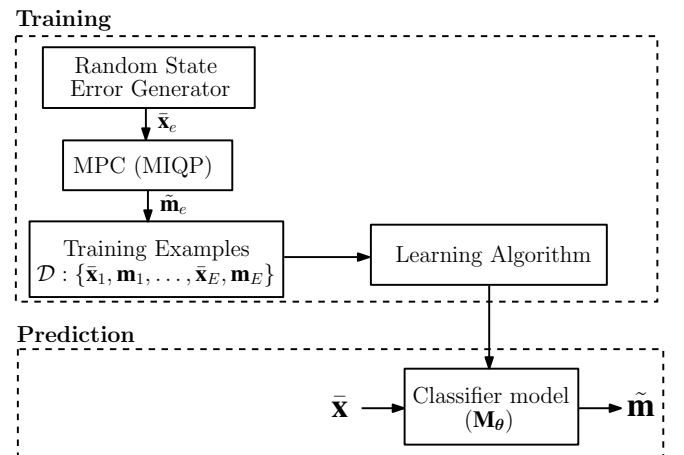


**Figure 11.** Supervised learning framework for mode schedule selection. A dataset of $E$ labelled datapoints is generated by solving Optimization Problem 1. From the training examples, a classifier is trained to return the mode schedule based on the state error vector.

This formulation lends itself well to a supervised learning setting, where the objective is to train a classifier model that can select an effective mode schedule given the error state.

**Optimization Problem 2**: Given current error state $\bar{\mathbf{x}}_0$, nominal trajectory $(\mathbf{x}_i^\star, \mathbf{u}_i^\star)$, and mode schedule $\mathbf{m}$, solve

$$\min_{\bar{\mathbf{x}}_i,\ \bar{\mathbf{u}}_i} \quad \bar{\mathbf{x}}_N^\mathsf{T} \mathbf{Q}_N \bar{\mathbf{x}}_N + \sum_{i=0}^{N-1} \left( \bar{\mathbf{x}}_{i+1}^\mathsf{T} \mathbf{Q} \bar{\mathbf{x}}_{i+1} + \bar{\mathbf{u}}_i^\mathsf{T} \mathbf{R} \bar{\mathbf{u}}_i \right)$$

subject to
$$\bar{\mathbf{x}}_{i+1} = \bar{\mathbf{x}}_i + h \left[ \mathbf{A}_i \bar{\mathbf{x}}_i + \mathbf{B}_i \bar{\mathbf{u}}_i \right],$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_0,$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_1 \quad \text{if } m_i \text{ specifies that } c \text{ is sticking,}$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_2 \quad \text{if } m_i \text{ specifies that } c \text{ is sliding left,}$$
$$\bar{\mathbf{u}}_{c,i} \in \mathcal{C}_3 \quad \text{if } m_i \text{ specifies that } c \text{ is sliding right.}$$

We present the learning framework used to design the classifier model shown in Fig. 11. Using the Optimization Problem 1 presented in Eq. (21), we generate a dataset of $E$ training example $\{\bar{\mathbf{x}}_e, \mathbf{m}_e\}$, where $\mathbf{m}_e$ represents the mode schedule associated with the $e^{th}$ datapoint. The purpose of the machine learning algorithm is to train a classifier model that minimizes the cross-entropy error function of the labelled training set. In this paper, we use a fully connected neural network as described in Table 1.

This new hybrid control architecture leads to a convex optimization program with a prescribed mode sequence. The main attraction of this approach, detailed in Algorith 1, is to convert a non-convex mixed-integer quadratic program into a convex quadratic program that can be solved in real-time.

# 7 NUMERICAL RESULTS

This section performs a numerical simulation study to evaluate the performance of the controller MPC-LMS introduced in Section 6 on a planar manipulation task. We consider the task of stabilizing an object with perturbed initial conditions about a nominal trajectory. We compare the performance of the proposed controller against three baselines controllers: MPC-MIQP, which is optimal with linearized dynamics, MPC with Sticking Contacts, and LQR with Frictionless Contact.

1. **MPC-MIQP**. The MPC controller with Mixed-Integer Quadratic Programming is described in Section 6.2 and represents the optimal baseline. This controller can only be executed in numerical simulations as it is not fast enough for online executions.
2. **Sticking Contacts**. To evaluate the value of a controller design that can reason across multiple contact modes, we compare the performance against a controller that is limited to reason over a single contact configuration: sticking. This controller requires solving Optimization Program 2 with the mode sequence fixed as $\mathbf{m} = \{S, \ldots, S\}$ for the entirety of the control horizon. This strategy is equivalent to that used by Posa et al. (2016), Woodruff and Lynch (2017), and Pardo et al. (2017).
3. **Frictionless Contacts**. When controlling systems with sustained contact interactions, one could assume a frictionless contact model that neglects tangential frictional forces. This assumption leads to a smooth dynamical system that can be stabilized using a Linear-Quadratic-Regulator (LQR). This controller can be interpreted as assuming a contact that can slide freely while applying positive normal forces.

This section investigate a comparison of the application of the MPC-LMS controller on a straight-line trajectory with perturbed initial conditions. We compare the sensitivity to initial state errors of the proposed algorithm against three benchmarks across 100 simulations with random initial perturbations. Finally, we test the robustness of the MPC-LMS controller to errors made in contact classification.

## 7.1 Straight-Line Tracking Simulation

In this section, we consider the task of controlling the motion of a square using a single point pusher about a straight line trajectory at a constant velocity shown in Fig. 12(a). The point pusher system is defined by the state vector $\mathbf{x} = [x\ y\ \theta\ p_c]^\mathsf{T}$ and the control vector $\mathbf{u} = [f_n\ f_t\ \dot{r}_y]$, where $r_y = \frac{a}{2}\tan\phi$. The initial conditions are $x_0 = -0.01$ m, $y_0 = 0.03$, $\theta_0 = 30$ deg, and $r_{y0} = -0.02$ m. The physical parameters used in the numerical simulation are reported in Table 3.

The classifier model used in the MPC-LMS controller is learned using a multilayer neural network with the architecture reported in Table 1. The controller design parameters used in the numerical simulations are $h = 0.03$ s, $N = 35$, $\mathbf{Q} = 10\ \text{diag}\{3, 3, 0.1, 0\}$, $\mathbf{Q}_N = 2000\ \text{diag}\{3, 3, 0.1, 0\}$, and $\mathbf{R} = 0.5\ \text{diag}\{1, 1\}$. The prediction horizon is split into $M = 8$ bins during which the contact modes are held constant. The number of time steps associated with each aggregated contact mode section $m_m$ is $\{1, 5, 5, 5, 5, 5, 5, 4\}$ with the associated contact mode weight matrix $\mathbf{W} = 0.1\ \text{diag}\{0, 3, 1, 1, 1, 1, 1, 1\}$ for all time steps. To regulate the velocity of the pusher, we include the pusher velocity constraints $|v_n| \leq 0.3$ m/s and $|v_t| \leq 0.3$ m/s to the optimization program. All MPC based benchmarks make use of identical design parameters following those of the MPC-LMS while the LQR based controller uses $\mathbf{Q} = 10\ \text{diag}\{3, 3, 0.1, 0\}$ and $\mathbf{R} = 100\ \text{diag}\{0.5, 0.5\}$.

In Fig. 12(a), we show the tracking performance of the MPC-LMS controller and compare its performance against the three baselines algorithms. Results show that the MPC-MIQP and the MPC-LMS algorithms succeed in tracking the desired trajectory. The frictionless controller is not able to track the nominal trajectory as its model neglects tangential frictional forces which causes the pusher to get stuck in a position offset from the nominal position. This limitation leads to the controller's inability to overcome friction and control the motion of the object. In contrast, the sticking controller is not able to track the trajectory as it does not have the ability to slide, a required skill to recover from the initial perturbations in $r_y$. In Fig. 13, the error dynamics of the simulation presented in Fig. 12(a) are shown to tend towards zero after approximately 2 s. In Fig. 14, we show the control effort of the simulation in Fig. 12(a). The control input space $\mathbf{u} = [f_n\ f_t\ \dot{r}_y]^\mathsf{T}$ is mapped to the robot end-effector velocity $\mathbf{v}_c$ using Eq. (14). Note that although the commanded velocities are discontinuous during contact switches, the resultant commanded robot positions $x_p$ and $y_p$ are smooth.

When the initial configuration of the pusher matches the nominal trajectory as in Fig. 12(b) with initial conditions $x_0 = -0.005$ m, $y_0 = -0.02$, $\theta_0 = 20$ deg, and $r_{y0} = 0$ m, the sticking configuration is able to track the nominal configuration. However, its limited control authority takes approximately 3 times longer to converge to the nominal trajectory. In practice, the controller's ability to slide the pusher relative to the object does not only add more control authority by allowing the pusher to move to a more strategic location relative to the object, but is also essential to return to the nominal contact location of the pusher on the
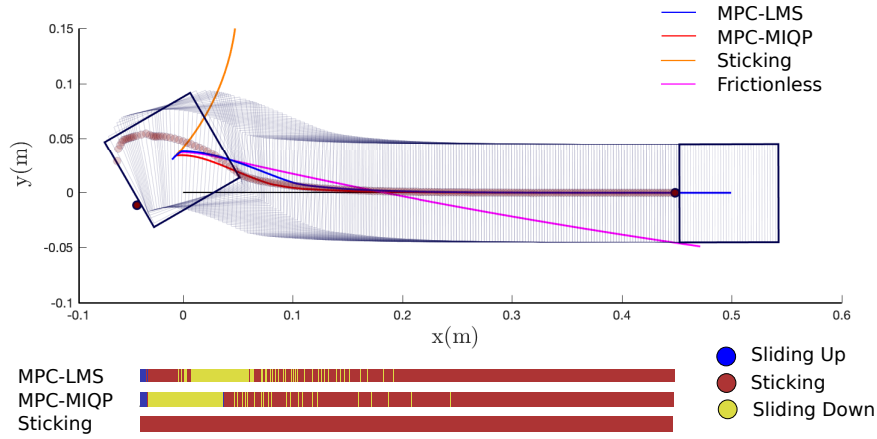


(a) Both the MPC-LMS and optimal baseline MPC-MIQP are able to track the trajectory. The frictionless controller that neglects friction is unable to slide to the proper contact location due to unexpected friction. The sticking controller is unable to to track the trajectory as it doesn't have the ability to slide.
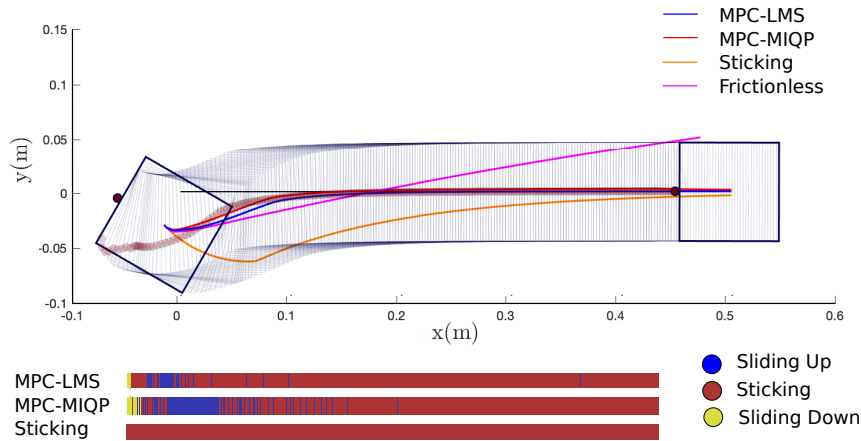


(b) When the initial configuration of the pusher matches that of the nominal trajectory, the sticking configuration is able to track the nominal configuration. However, its limited control authority leads to a significantly slower convergence to the nominal trajectory.

**Figure 12.** Closed-loop straight line tracking of a pushed square object. The MPC-LMS controller performance is compared to three benchmark: the optimal baseline MPC-MIPQ, MPC with sticking contacts, and LQR with frictionless contacts.
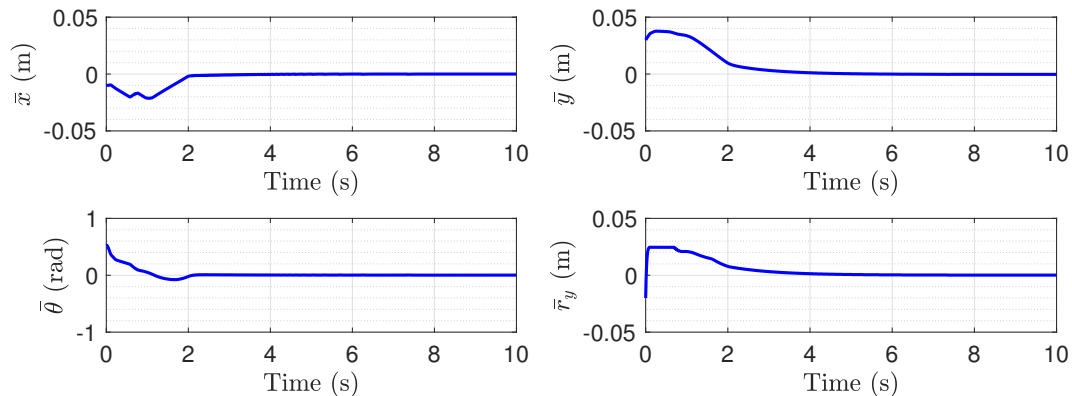


**Figure 13.** Error dynamics response from perturbed initial conditions associated with the MPC-LMS controller shown in Fig. 12(a). The perturbed initial conditions are shown to converge towards zero after approximately $2$ s.
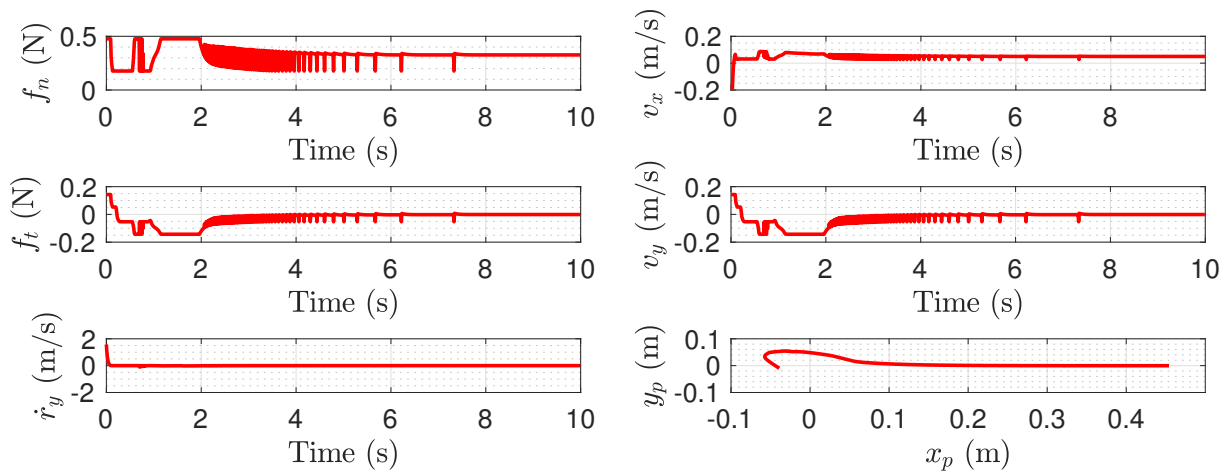
**Figure 14.** Control effort response from perturbed initial conditions associated with the MPC-LMS controller shown in Fig. 12(a). The control input space $\mathbf{u} = [f_n\ f_t\ \dot{r}_y]^\mathsf{T}$ is mapped to the robot end-effector velocity $\mathbf{v}_c$ using Eq. (14). Although the commanded velocities are discontinuous at contact switches, the resultant commanded robot positions $x_p$ and $y_p$ are smooth.

object following unexpected sliding, either caused by model uncertainty or external perturbations.

It is interesting to compare the contact modes enforced by the MPC-LMS vs. the optimal baseline MPC-MIQP. For example, in Fig. 12(a), while both sequences are not identical, they capture the same general behavior: quickly slide up, stick shortly, slide down, and stick during the remainder of the push. While the controller most often relies on sticking interactions to control the motion of the object, it exploits the additional sliding modes to increase the ability of the pusher to rotate the object. The inability of the sticking controller to recover from external perturbations shows that this ability is not only desirable for improved performance but also necessary to perform the task.

### 7.2 Sensitivity to initial state errors

To test the robustness of the MPC-LMS controller, we test its performance on 100 trajectory tracking simulation

with perturbed initial conditions drawn uniformly from the range $\pm[0.03\ 0.03\ .4\ 0.025]$ associated with the dimensions $[x\ y\ \theta\ r_y]$. Figure 15 shows the mean and variance of the euclidean distance between the measured and desired position over the entirety of the trajectories. Results show that the MPC-LMS achieves an average performance ($E = 0.0064 \pm 0.0055$ m) that is comparable to that of MPC-MIQP ($E = 0.0051 \pm 0.0051$ m). Both the frictionless and sticking control architectures have high error ($E = 0.0247 \pm 0.0063$ m) and ($E = 0.0499 \pm 0.055$ m) respectively, as the controllers are unable to track the nominal trajectory.

The major advantage presented by the MPC-LMS controller is it that is allows for real-time execution on an experimental setup. Results taken over 100 random simulations show that the MPC-LMS controller achieves an average computational time of $0.0028 \pm 0.00075$ s compared to that of $0.40 \pm 0.17$ s for the MPC-MIQP.

We parametrize the classifier model presented in Fig. 10 using a neural network, as reported in Table 1. Table 2



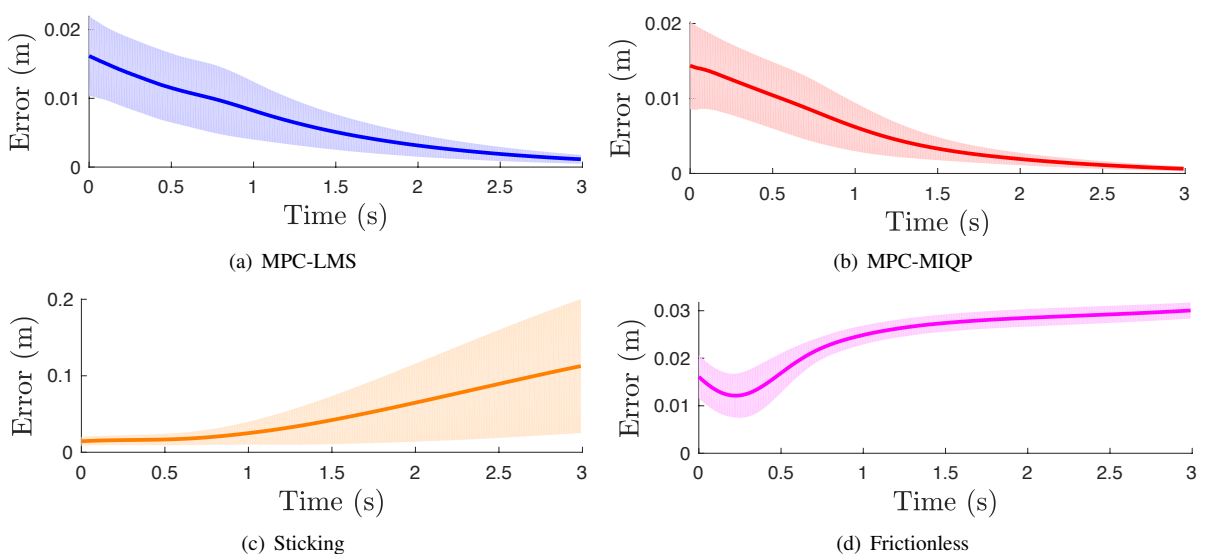(a) MPC-LMS

(b) MPC-MIQP

(c) Sticking

(d) Frictionless

**Figure 15.** Trajectory tracking error mean and variance taking over 100 simulations with random initial states. The error is compute as the euclidean distance between between the observed center of mass position and its desired location. In general, the sticking and frictionless controllers cannot control the system.

**Table 1.** Neural network parameters.

| Property | Value |
|---|---|
| Number of hidden layers | 3 |
| Neurons in hidden layer 1 | 32 |
| Neurons in hidden layer 2 | 50 |
| Neurons in hidden layer 3 | 50 |
| Activation functions | ReLu |
| Output layer | Softmax |
| Loss function | Cross entropy |

shows the prediction accuracy of the neural network trained on $100,000$ labelled data points on a validation set of $50,000$ labelled data points both generated by sampling the error state from a normal distribution with standard deviation $[0.03\ 0.03\ .4\ 0.025]$, associated with the dimensions $[x\ y\ \theta\ r_y]$. We evaluate the performance on each mode individually, as defined in Fig. 8. The prediction accuracy is above $90\ \%$ on all 8 contact mode clusters, showcasing the neural network's ability to accurately predict the contact mode based on the error state.

### 7.3   Sensitivity to contact mode errors

In Fig. 16, we explore the sensitivity of the MPC-LMS controller to mistakes committed by the contact mode classifier. Figure 16 investigates the performance of the controller when errors are randomly introduced in the contact mode selection relative to the MPC-MIQP optimal baseline.
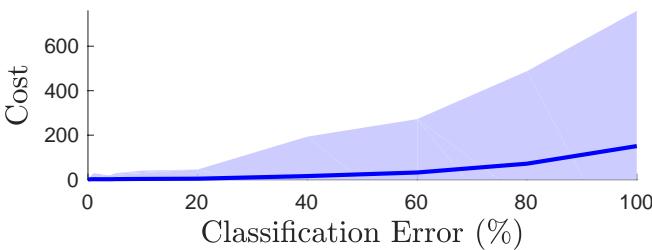


**Figure 16.** MPC cost vs. classification mistakes. Random classification errors are introduced on all contact modes to the MPC-MIQP solution to understand the sensitivity of the controller to classification mistakes.
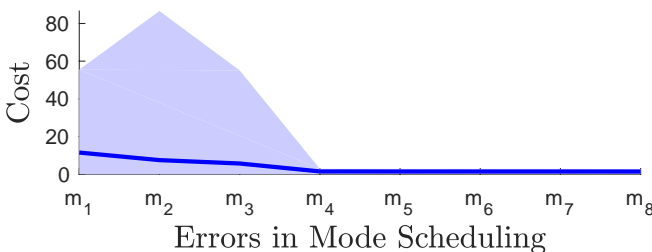


**Figure 17.** Tracking error vs. classification mistake at a given time step as defined in Fig. 8. Random classification errors are introduced ($100\%$ error) on individual contact modes to the MPC-MIQP solution to understand the sensitivity of the controller. Results show that committed mistakes during the beginning of the control horizon have the most important impact on controller performance.

Results show that the MPC controller degrades linearly up to a classification error of $20\%$, at which point the performance degrades significantly. Note that performance of the learned classifier described in Table 2 are well below this threshold.

In Fig. 17, we explore the sensitivity to classification errors as a function of the time step of the MPC program at which it was committed. As expected, results show that mistakes committed during the beginning of the control horizon have the most important impact on controller performance. We hypothesize that this is the case as errors committed during the beginning of the horizon have a stronger impact on the future trajectory of the object. Moreover, mistakes committed during the first time step are expected to have a more significant influence on the performance as MPC only applies the first command of the control sequence.

**Table 2.** Accuracy results for the straight line point pushing numerical results. The neural network predictions are evaluated on a validation set of 50K labelled data points. We evaluate the performance on each mode separately, as defined in Fig. 8.

| Mode Id. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.93 | 0.93 | 0.92 | 0.91 | 0.90 | 0.90 | 0.97 | 0.99 |

## 8   EXPERIMENTAL RESULTS

We experimentally validate the MPC-LMS controller design on a planar experimental setup using an industrial robotic manipulator (ABB IRB 120) shown in Fig.1. We refer the readers to the attached video for a visualization of the closed-loop pushing experiments. The pose of the pushed object is tracked using a Vicon system (Bonita). The experimental setup is depicted in Fig. 1, where a metallic rod (pusher) attached to the robot is used to push an aluminum object on a flat surface (plywood, abs, delrin, etc.).

Section 8.1 considers a trajectory tracking problem using a square object with a point robotic pusher while Section 8.2 extends the contact configuration to a two point pusher. The objective of the controller is to track the trajectory of the center of mass of the object along the race track defined by two circles of radii $0.15$ m at a constant velocity of $v = 0.05$ m/s, as in Fig. 18.

**Table 3.** Experimental system parameters.

| Property | Symbol | Value |
|---|---|---|
| Coefficient of friction (pusher-object) | $\mu_p$ | 0.3 |
| Coefficient of friction (object-table) | $\mu_g$ | 0.35 |
| Mass of object, $kg$ | m | 0.827 |
| Object side length, $m$ | a | 0.09 |
| Line pusher width, $m$ | d | 0.03 |

(a) Point pusher tracking a trajectory for 7 consecutive laps. The black line represents the desired trajectory and the blue lines track the center of mass of the object.



(b) Point tracking of the same trajectory with external perturbations for a single lap. The black line represents the desired trajectory and the hand represents the locations and directions in which the perturbations were applied.
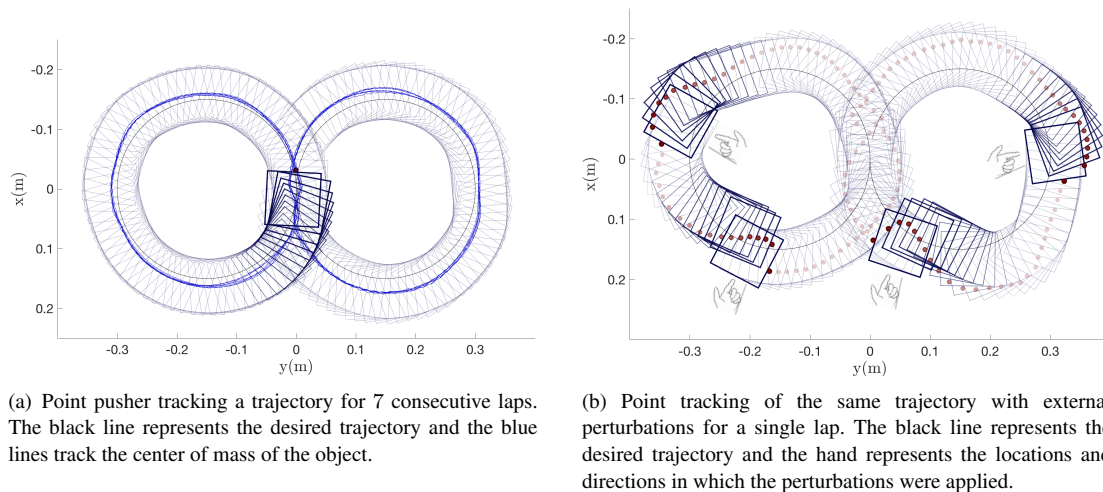
**Figure 18.** Experimental tracking of the $\infty$ track with the MPC-LMS controller for the robotic point pusher.

## 8.1 Case Study A: Single Point Pushing



**Figure 19.** Experimental setup for point pusher.

Figure 19 shows the experimental setup for the planar manipulation task with a point pusher. The desired trajectory is set to $0.05$ m/s and all control parameters are kept identical to those described in Section 7, and we evaluate the performance of the MPC-LMS controller design approach on the race track in Fig. 18. First, we consider the ability of the controllers to track the nominal trajectory without exerting any external perturbations on the system. Second, we perform the experiments while subjecting the pusher to controlled external perturbations to evaluate the reactive capabilities of the controller.

Table 5 shows the prediction accuracy of the neural network trained on $120,000$ labelled data points on a validation set of $50,000$ data points both generated by sampling the error state from a normal distribution with standard deviation $[0.03\ 0.03\ .4\ 0.025]$, associated with the dimensions $[x\ y\ \theta\ \phi_c]$. The system parameters used in the experiments are presented in Table 3.

**Table 4.** Accuracy results for the point pushing experimental results on a $8$ track trajectory. The neural network predictions are evaluated on a validation set of 50K labelled data points. We evaluate the performance on each mode separately, as defined in Fig. 8.

| Mode Id. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.95 | 0.95 | 0.95 | 0.93 | 0.95 | 0.96 | 0.96 | 0.98 |

*8.1.1 Point Pusher Experiments* Figure 18 shows the robot manipulator pushing the square object about the race track without any external perturbations for 7 consecutive laps. The black line is the desired trajectory and the blue line tracks the geometric center of the object. The MPC-LMS controller succeeds in tracking the desired trajectory within an average state error of $0.02$ m. The stability of the MPC-LMS can be seen in Fig. 18(a) by the small variance associated with the 7 consecutive trajectories that overlap within a very small tolerance.

In Fig. 18(b), we apply four successive impulsive forces in the transverse motion to the object to perturb the system about its nominal trajectory and evaluate the stabilizing capabilities of the feedback controller. The forces are applied to ensure that the object is pushed at the same location by an equal distance on each experiment. The controller quickly reacts to external perturbations and acts in such a manner to eliminate the perturbation. During real-time executions, the controller reasons about both future control inputs and contact modes to eliminate errors at an average frequency of $250$ (Hz).

## 8.2 Case Study B: Pushing with Line Contact



**Figure 20.** Experimental setup for line pusher.

The experimental setup for the planar manipulation task with a line pusher is shown in Fig. 20. The task of the controller is to determine the motion of the robot in real-time to control
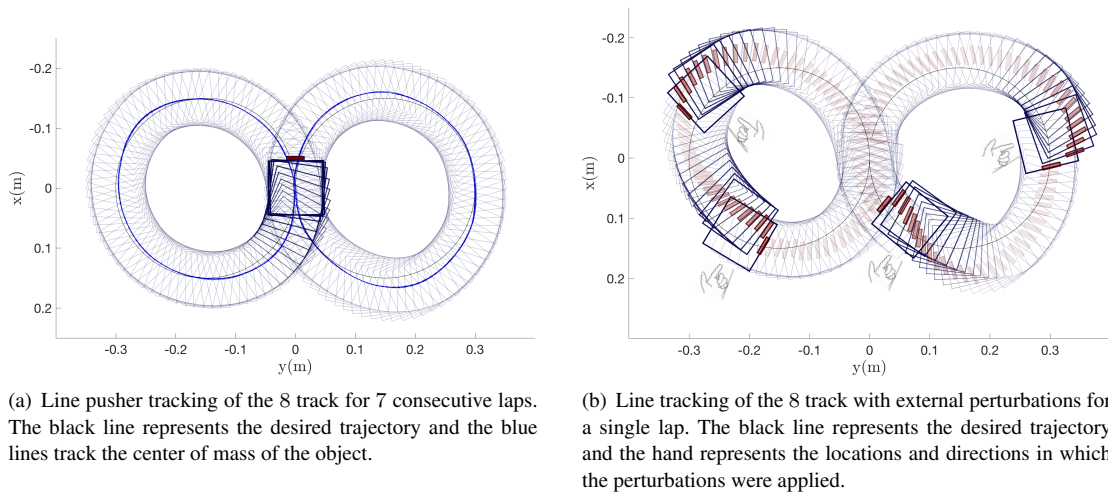
(a) Line pusher tracking of the 8 track for 7 consecutive laps. The black line represents the desired trajectory and the blue lines track the center of mass of the object.

(b) Line tracking of the 8 track with external perturbations for a single lap. The black line represents the desired trajectory and the hand represents the locations and directions in which the perturbations were applied.

**Figure 21.** Experimental tracking of the 8 track with the LMS controller for the robotic point pusher.

the motion of the object about a race track shaped trajectory defined by two circles of radii $0.15$ meters at a constant velocity of $v = 0.05$ m/s. We model the line pusher as 2 contact points that are constrained to move as a rigid-body, with the position of the center point of pusher denoted as $r_y$ and state vector defined by $\mathbf{x} = [x\ y\ \theta\ r_y]^\mathsf{T}$, only one sliding velocity is considered as both points move as a rigid body. We include the sliding velocity of only one point pusher, since both points move as a rigid body.

Following a similar approach to that described in Section 8.1, we train a classifier model using $80$ K labelled data points to predict the optimal mode schedule based on the error state of the system. The physical properties used to parametrize the classifier model and the neural network properties are related in Tables 1 and 3, respectively.

For the line pusher, both contact points are constrained to have the same contact mode as they are constrained to move together as a rigid body and remain in contact with the object. The controller design parameters are selected as $h = 0.03$ seconds, $N = 35$, $\mathbf{Q} = 10$ diag$\{1, 1, 1, 0.1\}$, $\mathbf{Q}_N = 2000$ diag$\{1, 1, 1, 0.1\}$, and $\mathbf{R} = $ diag$\{1, 1, 1, 1, 0.01\}$. We split the prediction horizon into $8$ parts during which the contact modes are held constant. The number of time steps associated with each contact mode section $m_m$ is $\{1, 5, 5, 5, 5, 5, 5, 4\}$ with the associated weight matrix $\mathbf{W} = 0.1$ diag$\{0, 3, 1, 1, 1, 0, 0, 0\}$ for all time steps.

*8.2.1 Line Pusher Experiments* Figure 21(a) depicts the robotic line pusher pushing the square object about the race track without any external perturbations for 7 consecutive laps. Figure 21(b) depicts the robotic line pusher pushing the square object about the race track with external perturbations for a single lap. Each time a perturbation is encountered,

**Table 5.** Accuracy results for the line pushing experimental results on a $8$ track trajectory. The neural network predictions are evaluated on a validation set of 50K labelled data points. We evaluate the performance on each mode separately, as defined in Fig. 8.

| Mode Id. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Accuracy | 0.96 | 1.0 | 0.95 | 0.98 | 0.99 | 1.0 | 1.0 | 1.0 |

the pusher reacts to reduce the error by following a fast sliding motion to stabilize the object and then push it back towards the desired trajectory using a sticking phase. The controller is executed online at an average frequency of 200 Hz. Note that the frequency is lower because of the extra complexity associated with the larger input space and consequent additional constraints.

## 9 Influence of Control Parameters

In this section, we investigate the dependence and sensitivity of the proposed controller to design parameters. Specifically, we evaluate the effect of the controller frequency, tracking velocity, planning horizon, estimated coefficient of friction, and radius of curvature of the track on the closed-loop performance of the pusher-object system.

We evaluate the performance of the controller by computing the average mean squared error between the desired trajectory and the actual trajectory for the point pusher system. All experiments are conducted by tracking three consecutive laps of the race track at $0.08$ m/s, with the last two laps subject to two controlled external perturbations. An example of this can be viewed in the video attachment.

### 9.1 Controller Frequency

The pusher-object system is naturally unstable in its forward motion and can be stabilized using a feedback controller for trajectory tracking. A natural question to ask is "what is the lowest controller bandwidth at which the system can be controlled?" Figure 22(a) depicts the effect of the control frequency on the closed-loop tracking performance. Results show that the system requires a minimal control bandwidth of 20 Hz for closed-loop stability. Above this frequency, the performance remains unchanged. Note that, intuitively, this minimum control frequency will change with the velocity of the target trajectory.

### 9.2 Tracking Velocity

The model used for controller design uses the quasi-static approximation, which has been shown to be a good approximation for object velocities under $0.05$ m/s by Bauza
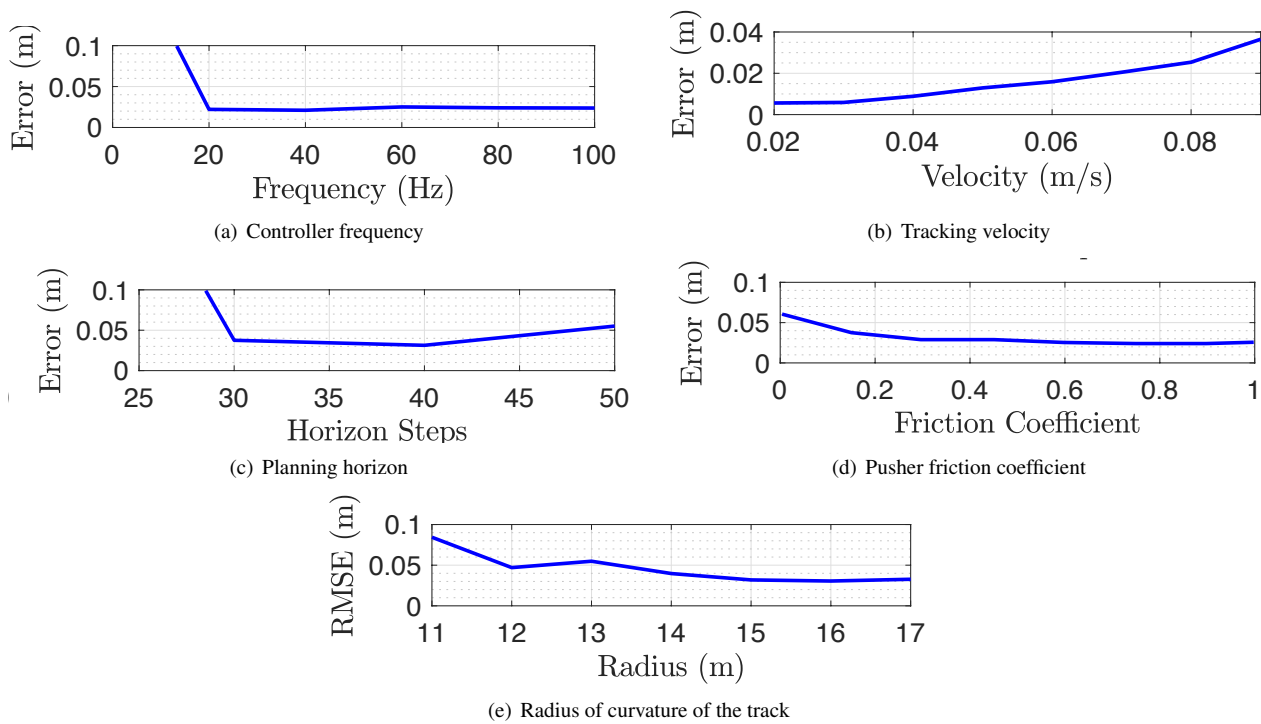
(a) Controller frequency

(b) Tracking velocity

(c) Planning horizon

(d) Pusher friction coefficient

(e) Radius of curvature of the track

**Figure 22.** Experimental analysis of the performance as a function the a) controller frequency, b) tracking velocity, c) planning horizon, d) error in coefficient of friction, and e) radius of curvature of the track.

and Rodriguez (2017). Above this velocity, the unmodeled inertial forces are appreciable and can negatively impact performance. In Fig. 22(b), the controller performance degrades with the increase of the desired velocity. The controller remains stable until 0.1 m/s at which point the system becomes unstable. We hypothesize that the tracking error grows with the tracking velocity as inertial forces start to have an impact on the object motion and are unaccounted for in the controller design. Furthermore, trajectory tracking tasks at higher velocities are more difficult as they require shorter reaction times to correct mistakes.

### 9.3 Planning horizon

Due to the underactuated nature of the dynamics of the pusher-object system, a certain planning horizon length must be considered for closed-loop stability. The controller must reason beyond instantaneous actuation since the forces required to drive the task in the direction of the goal might not be feasible at the current instant. To investigate the influence of the planning horizon, we gradually increase the planning horizon from 5 to 50 time steps, which corresponds to planning horizons of 0.15 to 1.5 seconds. Results in Fig. 22(c) show that the system requires a minimal planning horizon of 20 steps for stability, while performance continues to increase until it reaches its peak performance around 35 time steps. Beyond that, the controller performance degrades, possibly due to the additional computational complexity associated with planning for very long horizons.

### 9.4 Coefficient of Friction

Coefficients of friction of robotic systems are among the most difficult parameters to estimate and can have a notable impact on the system's dynamics. Often it is difficult or

impossible to estimate without explicit sensing force Fazeli et al. (2018). Here, we investigate how the performance of the controller is impacted by varying our estimate of the coefficient of friction between the pusher and the object from 0 to 1. Our best estimate of this value, based on steel-steel contact interactions, is detailed in Table 3 as 0.3. Figure 22(d) shows that the performance increases monotonically with the estimate of the coefficient of friction.

At first glance, these results seem surprising and counter intuitive as it is unreasonable that the real value of the coefficient of friction lies as high as 1. We hypothesize that the controller benefits from overestimating the pusher-object coefficient of friction as it leads to more aggressive robot motions, where the robot favors high tangential velocity during sliding motion to ensure that the reaction force lies on the boundary of the friction cone.

### 9.5 Race Track Radius of Curvature

Finally, in Fig. 22(e) we test the performance of the controller design by increasing the desired radius of curvature of the race track trajectory from 11 to 17 cm. As expected, the performance of the controller degrades as the radius of curvature decreases because tighter curves require more aggressive pusher motions with higher control authority and are more difficult to track. The minimal radius of curvature achieved is 11 cm prior to the controller going unstable while the maximum radius of curvature is constrained to the kinematic workspace of the robotic arm. The minimal radius of curvature achieved is executed with a square object with side lengths of 9cm.

## 10   CONCLUSION

In this work, we present a feedback controller for a planar pushing task. The control formulation is based on a Model Predictive Control approach, where the hybridness and underactuation associated with contact are explicitly enforced as linear constraints within a mixed-integer optimization program. We propose an online approximate solution MPC-LMS to the offline optimal control problem to achieve real-time computational requirements while reasoning across multiple contact modes. The proposed MPC-LMS approach consists in formulating the search for optimal modes offline separately from the search for optimal control inputs online, by leveraging machine learning methods to select mode sequences from prior experience.

We validate the MPC-LMS algorithm through numerical simulations and compared to the optimal baseline as well as two other benchmarks (pure sticking and pure sliding). We further validate our controller design experimentally, where the feedback controller successfully stabilizes the motion of a sliding object about various nominal trajectories. We demonstrate implementation of the controller for two manipulation tasks (point and line pushers) while providing a framework for the mechanics that generalizes to multiple contact formations and different object shapes. Finally, we study the changes in the performance of the algorithms for controlled variations in the control frequency, object velocity, predictive horizon length, and radius of curvature of the trajectory.

## 11   Aknowedgements

## 12   Funding

### References

Alessandro Alessio and Alberto Bemporad. Feasible mode enumeration and cost comparison for explicit quadratic model predictive control of hybrid systems. *IFAC Proceedings Volumes*, 39(5):302–308, 2006.

M. Bauza and A. Rodriguez. A probabilistic data-driven model for planar pushing. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3008–3015, 2017.

A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407 – 427, 1999.

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

Martin Buehler, Daniel E Koditschek, and Peter J Kindlmann. Planning and control of robotic juggling and catching tasks. *The International Journal of Robotics Research*, 13(2):101–118, 1994.

N. Chavan-Dafle, A. Rodriguez, Bowei Tang R. Paolini, Siddhartha Srinivasa, Michael Erdmann, M.T. Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014.

Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *Robotics: Science and systems VII*, 1, 2011.

Mehmet R Dogar and Siddhartha S Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3):217–236, 2012.

Nima Fazeli, Russ Tedrake, and Alberto Rodriguez. Identifiability analysis of planar rigid-body frictional contact. In *Robotics Research*, pages 665–682. Springer, 2018.

S. Goyal, A. Ruina, and J. Papadopoulos. Wear. *Planar sliding with dry friction Part 1. Limit surface and moment function*, 143:307 – 330, 1991.

Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015. URL http://www.gurobi.com.

F.R. Hogan and A. Rodriguez. Feedback control of the pusher-slider system: a story of hybrid and underactuated contact dynamics. In *Proceedings of the 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, USA, December 18 – 20., 2016.

F.R. Hogan, E.R. Grau, and A. Rodriguez. Reactive planar manipulation with convex hybrid MPC. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2018.

Yifan Hou, Zhenzhong Jia, and Matthew T Mason. Fast planning for 3d any-pose-reorienting using pivoting. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1631–1638. IEEE, 2018.

Michael C Koval, Nancy S Pollard, and Siddhartha S Srinivasa. Pre-and post-contact policy decomposition for planar contact manipulation under uncertainty. *The International Journal of Robotics Research*, 35(1-3):244–264, 2016.

Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

Mircea Lazar, WPMH Heemels, Siep Weiland, and Alberto Bemporad. Stabilizing model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 51(11): 1813–1818, 2006.

S.H. Lee and M. Cutkosky. Journal of Manufacturing Science and Engineering. *Fixture planning with friction*, 113(3):320 – 327, 1991.

K.M. Lynch and M.T. Mason. Stable pushing: mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533 – 556, 1996.

K.M. Lynch, H. Maekawa, and K. Tanie. Manipulation and active sensing by pushing using tactile feedback. In *Intelligent Robots and Systems (IROS), 1992 IEEE/RSJ International Conference on*, 1992.

M.T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53 – 71, 1986.

M.T. Mason. *Mechanics of robotic manipulation*. MIT Press, Cambridge, Massachusetts, 2001.

Todd D Murphey and Joel W Burdick. Feedback control methods for distributed manipulation systems that involve mechanical contacts. *The International Journal of Robotics Research*, 23 (7-8):763–781, 2004.

G.L. Nemhauser and L.A. Wolsey. *Integer Programming and Combinatorial Optimization*. Wiley, Chichester, England, 1988.

Richard Oberdieck and Efstratios N Pistikopoulos. Explicit hybrid model-predictive control: The exact solution. *Automatica*, 58: 152–159, 2015.

Diego Pardo, Michael Neunert, Alexander W Winkler, Ruben Grandia, and Jonas Buchli. Hybrid direct collocation and control in the constraint-consistent subspace for dynamic legged robot locomotion. In *Robotics: Science and Systems*, 2017.

M. Posa, C. Cantu, and R. Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69 – 81, 2014.

M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016.

Gerrit Schultz and Katja Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on mechatronics*, 15(5):783–792, 2009.

D.E. Stewart and J.C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673 – 2691, 1996.

Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems (RSS)*, 2018.

Andrés Klee Valenzuela. *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain*. PhD thesis, Massachusetts Institute of Technology, 2016.

J.Z. Woodruff and K.M. Lynch. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017.

J. Zhou, R. Paolini, J.A. Bagnell, and M.T. Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016.

J. Zhou, James Bagnell, and Matthew Mason. A fast stochastic contact model for planar pushing and grasping: theory and experimental validation. In *Robotics Science and Systems, Cambridge, MA, USA, July 12–16*, 2017.

Jiaji Zhou and Matthew T Mason. Pushing revisited: Differential flatness, trajectory planning and stabilization. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2017.