Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

A Learning Approach To The FOM Problem

Eudald Romo Grau

Supervised by Alberto Rodriguez Garcia and Maria Alberich Carramiñana

April, 2017

Thanks to Alberto Rodriguez for tutoring my research, for providing me with the required tools and financial support to undertake this project and for hosting me in his laboratory; to Maria Alberich for supervising and tutoring this thesis; to Francois Hogan for sharing his ideas on family of modes trajectory control and for introducing me to his previous work, to Maria Bauza for sharing with me her insights in machine learning techniques; to Nikhil Chavan-Daffle for sharing with me his insights in complementarity constraint problems; to all the members of the MCube Lab for sharing their thoughts on my research, to Centre de Formacio Interdisciplinaria Superior for offering me the possibility and the financial support to take part in this project, to Massachusetts Institute of Technology for providing the required facilities required to develop this project and to Generalitat de Catalunya for their financial support.

Abstract

The family of modes (FOM) approach to solving model predictive control (MPC) problems is a novel heuristic technique developed at MIT to solve the time complexity of traditional MPC solving techniques. This study addresses some of the issues associated with the previous formulations of this technique by increasing the sequential robustness of the FOM and providing methodologies to choose the parameters required for the heuristic. A general simulation interface is developed together with techniques to score and compare the obtained trajectories. Then an statistics and machine learning based methodology to tune the parameters is proposed and the results are compared with the original ones. Finally, experimental procedures are developed to validate the results.

Keywords

Manipulation, Online Control, MPC, FOM, Underactuation, Hybridness

1. Goal of this study

Hogan [14] recently provided an heuristic technique called family of modes (FOM) to solve model predictive control (MPC) problems under hybrid constraints and underactuation. The goal of this study, that will be revisited and expanded in section 4, is to further develop this new method and to expand its usage in the robotics manipulation community.

In order to do that, we reinforce the method by fixing some of it's weaknesses -as it's sequential properties-, we provide comparison tools to test the method against traditional hybrid MPC solving techniques -as mixed integer quadratic programming (MIQP)-, and to provide simple and systematic techniques to optimize a general MPC problem.

To proof the scalability of this method we increase the complexity of the systems beyond the problem used in Hogan's work -tracking straight lines with of a single point pusher and a square slider.

Some simple control policies as proportionalintegralderivative (PID) controllers are able to control systems that scape the model they were designed for. Inspired by this fact, we check the robustness of the FOM method by using it in models that are different from those it is designed for.

In section (TODO: add conclusion/discussion section) we discuss the new lines of research we plan on exploring before presenting this study to 2017's International Symposium on Robotics Research (ISRR). We hope that the results from our studies will increase the performance and simplify the implementation of FOM and will bring it closer to the robotics manipulation research community by both providing systematic set-up tools and showing it's robustness and scalability.

In the following sections we introduce the concepts of manipulation problems under hybridness required to explain the FOM method and show its motivation.

2. Planar Pushing

As Mason explained in his talk during his CITRIS People and Robots Seminar Series talk [21], robotics manipulation research has been traditionally based on a pick and place methodology that consisted in solving two fundamental problems:

1. **Determine a stable grasping configuration:** A stable grasping configuration was generally obtained by finding a set of points that completely constrained the picked object position and orientation with respect to the gripper.

2. **Solving a collision-free trajectory planning problem:** Once the object was grasped it was rigidly attached to the robot gripper and the trajectory planning problem was completely determined at every point.

This methodology was constantly improved as the manipulation community considered broader and more complex problems, as Rodriguez' and Mason's study [25], that generalized the concept of stable grasp to caging -a partial immobilization of the grasped object that allows it to move in a bounded space. Other modifications added further constraints to the problem, as the concept of compliant grasping manipulation [19], where the object motion of the object is constrained by the task and the constraint is enforced without exerting heavy force loads on the object. An example of compliant manipulation the process of activating a lever. The lever only has one degree of freedom and can easily be broken by small

planning uncertainties if a powerful robotic arm is used and the trajectory is controlled uniquely with the robot position without considering the applied forces. It should be noted that all these variations were applied to the same pick and place grasping problem. Some of the main motivations that made stable grasping so popular are:

- **Easy to track:** The obtained solutions are constrained by the contact points and the gripper pose so, even if the object could move inside the gripper (as in cagings) the object position could be deterministically bounded during all the trajectory.

- **Robustness:** Most grasping point predictions are robust to small sensor noise and applying enough grasping force allows ignoring uncertainties in the grasped object friction coefficients.

- **Simplicity:** As explained, the manipulation problem could be formulated simply by a grasping point identification and a path planning problem.

At least as far as on 1980's, some drift towards the study of the mechanics of planar pushing started to appear with studies as Mason's [20]. The main motivations for this drift were to handle problems that were not tractable with grasping or to better understand the grasping mechanics. A non-exhaustive list of the motivations of planar pushing could be:

- **You don't need to lift:** Lifting heavy objects using a traditional robotic arm can put heavy torque strain on the arm in certain arm configurations and the strain can break or topple the arm. If most or all of the planned trajectory can be done through pushing, the required force is usually reduced and when the maximum force is achieved safety mechanisms can be implemented so that the arm doesn't break.

- **Sometimes you need to push to grasp:** From the observation of human object handling, it can be seen that sometimes we prefer or need a previous pushing manipulation before grasping an object. As can be observed in Figure 4, constrained spaces like a shelf may force us to previously push a book against one of the self walls to pull it outside so that we can grasp it.

- **It Can Represent In-Hand Re-grasping:** Humans constantly show the ability and need to modify a grasping configuration without letting the object out of the grasp. When we write with a pen or drink from a bottle, we are constantly adapting our initial grasp without letting the object go to achieve more comfortable configurations. Some studies as Chavan's [6] [16] have formulated this in-hand re-grasping as a planar pushing problem by considering either one moving finger or an external surface as the pusher and the contact between the fixed fingers and the object is described as planar sliding.

- **Weaker Quasistatic Assumptions:** As will be further explained in section 5.2, in order to use quasistatic grasping analysis (commonly done in the manipulation community) a stronger set of assumptions has to be taken than to use quasistatic pushing.

The mechanics of planar pushing are complex and contain some inherent qualities as underactuation and hybridness that need to be addressed in order to solve planning, simulating or control problems. In the following sections we are going to introduce the basic assumptions taken and models used in planar pushing, together with the difficulties that arise from them.

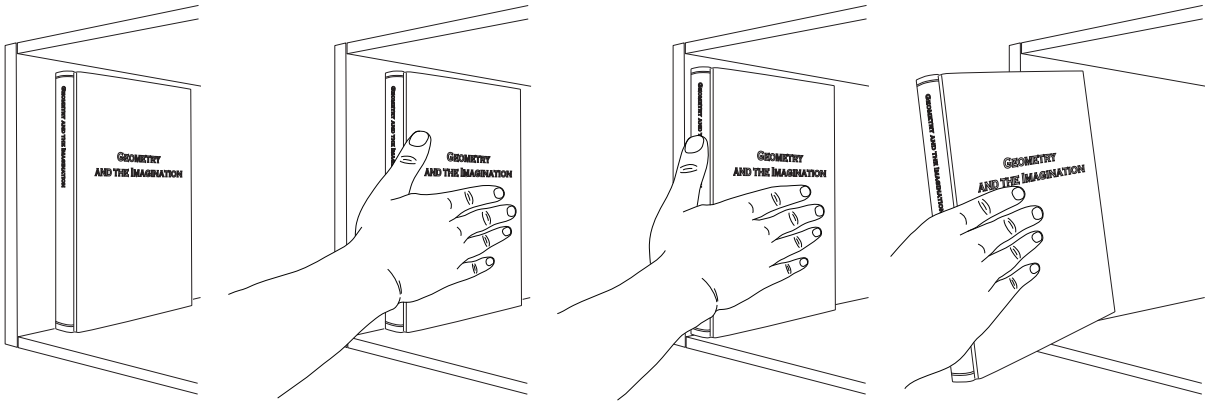

Figure 1: Figure from [14] that shows a human-like book grasping procedure.

2.1 Coulomb Friction

One of the usual assumptions in the robotics community is considering dry Coulomb contact forces. This model separates friction into two possible states (static and dynamic or kinetic friction) and the set of rules which describe it are a combination the work of Amontons [3] and Coulomb [8]:

- Amontons's first law: The force of friction is directly proportional to the applied load.
- Amontons's second law: The force of friction is independent of the apparent area of contact.
- Coulomb's law: Kinetic friction is independent of the sliding velocity.

This laws are usually condensed by defining the static (μ_s) and kinetic (μ_k) coefficients and considering the normal (f_n) and tangential (f_t) components of the contact force:

Theorem 2.1. Coulomb's Law:

$$f_t \begin{cases} < \mu_s \cdot f_n \ \textit{(for static contact)} \\ = \mu_k \cdot f_n \ \textit{(for kinetic contact)} \end{cases}$$

Where f_t is determined in the first situation by force balancing.

Remark 2.2. Through the rest of this thesis we use the common assumption taken in robotics manipulation of assuming a single coefficient of friction. Thus the previous theorem is simplified to $f_t \leq \mu \cdot f_n$

Mason cites prior statements by Da Vinci (from Truesdell's notes [27]) and experimental verification by Truesdell and Guillmor [11] and I would like to add the work of Euler [22], who first distinguished between static and kinetic friction.

2.2 Contact Slider-Support Surface

In planar pushing, there are two different contacts to be considered: the contact between the object (slider) and the support surface and the contact between pusher and slider. In this section we are going to introduce

the main theoretical results about the first one. We will start with Mason's study of the relation between friction force and both motion and instantaneous center of rotation (which will be used again in section 5.2) and we will end with Goyal's Limit Surface, that will provide a geometric interpretation to this relation.

The contact between the slider and the support surface is inherently complex. If the slider is supported by more than three contact points the distribution of support normal forces is indeterminate and, if Coulomb contact is assumed, this indetermination is propagated to the friction forces. The results in this section are based on the previous assumption that the pressure distribution is known (for our study we will consider a homogeneous distribution).

Assuming a known pressure distribution and considering $\mathbf{r}$ the position vector of a point of the slider, $\mathbf{v(r)}$ it's velocity vector, and p($\mathbf{r}$) the pressure at that point, then the total force and moment due to friction can be expressed as:

Definition 2.3. Total force due to friction:

$$\mathbf{f}_f = -\mu \int_A \frac{\mathbf{v(r)}}{|\mathbf{v(r)}|} p(\mathbf{r}) dA \tag{1}$$

Definition 2.4. Total moment due to friction:

$$n_f = -\mu \int_A \mathbf{r} \times \frac{\mathbf{v(r)}}{|\mathbf{v(r)}|} p(\mathbf{r}) dA \tag{2}$$

If the slider trajectory is assumed to be a rotation over a instantaneous center $\mathbf{r}_{IC}$, the velocity can be expressed as $\mathbf{v(r)} = \omega \times (\mathbf{r} - \mathbf{r}_{IC}) =$ and the previous results can be rewritten as:

Definition 2.5. Total force due to friction:

$$\mathbf{f}_f = -\mu \, sgn(\dot{\theta}) \, \hat{\mathbf{k}} \times \int_A \frac{\mathbf{r} - \mathbf{r}_{IC}}{|\mathbf{r} - \mathbf{r}_{IC}|} p(\mathbf{r}) dA \tag{3}$$

Definition 2.6. Total moment due to friction:

$$n_f = -\mu \, sign(\dot{\theta}) \int_A \mathbf{r} \cdot \frac{\mathbf{r} - \mathbf{r}_{IC}}{|\mathbf{r} - \mathbf{r}_{IC}|} p(\mathbf{r}) dA \tag{4}$$

Remark 2.7. These equations are well defined when the magnitude of $\mathbf{r}_{IC}$ tends to infinity and they correspond to pure translations perpendicular to it.

These equations provide a relation force-motion for known continuous pressure distributions but if non-zero support force is allowed at discrete points, they become undefined for rotations about each of them. In fact, the relation between motion and frictional force cannot generally be described as an analytical function. Goyal et al. [12] [13] applied concepts of classical plastic theory to describe planar sliding, deriving the concepts of limit surfaces for dry contact manipulation, that describe a mapping between motion and force.

Goyal's work can be generalised to friction models which are more general but, for clarity, the main concepts will be introduced with the special case of isotropic Coulomb friction (as was presented in their original paper).

Consider a rigid body sliding over a surface with a known or pressure distribution p($\mathbf{r}$) at every contact point and assume that each point of contact follows isotropic Coulomb friction law. Let $\mathbf{f(r)}$ be the differential frictional force applied by each slider particle to the surface. Coulomb law allows us to obtain two important properties:

1. **Limit Curve:** The set of possible point differential friction forces is contained inside a circle of radius $\mu p(\mathbf{r})$ centred at the origin of the force space. We will call this circle the limit curve (LC) of each contact point.

2. **Maximum Power Inequality.** The following relation is held at every contact point:

$$\forall \mathbf{f(r)}^* \in LC : \ (\mathbf{f(r)} - \mathbf{f(r)}^*) \cdot \mathbf{v(r)} \geq 0$$

   Where $\mathbf{v(r)}$ is the point velocity at each contact point and $\mathbf{f(r)}^*$ is an arbitrary force that can take any possible value inside the LC.

The maximum-power inequality and limit curve are concepts introduced from classical plastic theory that can be interpreted as follows: when slip occurs $\mathbf{v(r)}$ is non-zero and, for the case of a circular LC, the maximum power inequality restricts $\mathbf{f(r)}$ to have the same direction as $\mathbf{v(r)}$ and lie on the LC boundary. That is, the frictional load is on the limit curve and the motion vector is normal to the limit curve at $\mathbf{f(r)}$. When the point velocity is zero, the frictional load is undetermined.

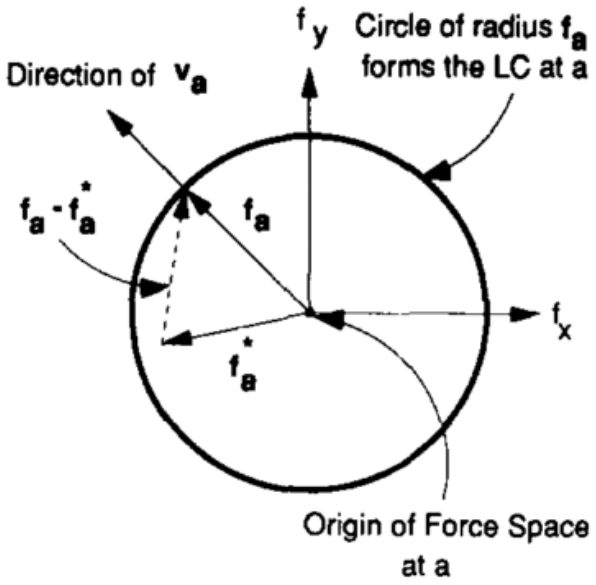

Figure 2: Figure from [12] that shows a limit curve for isotropic Coulomb contact at a certain position r_a. A unique set of v_a and f_a is determined when the point is slipping and it corresponds to the points in the circumference. When the force lies inside the circle, the point does not move and the frictional force is not determined.

Goyal's study further uses classical results of plastic theory to generalize this concept to general closed LC that are convex (to allow the maximum power inequality to be held at every contact point) and that enclose the origin of the force space (to allow positive energy dissipation). The motion-force relationship properties obtained in this general LC study are summarized in Figure 3 and explained here:

1. If $\mathbf{f(r)}$ is in the interior the LC, then $\mathbf{v(r)}$ is zero.

2. If **f(r)** is on a LC boundary point with a well-defined normal, then **v(r)** is perpendicular to the LC.
3. If **f(r)** is on a LC vertex, the direction of **v(r)** is not determined.
4. For all **f(r)** in a same flat region of the LC, a common **v(r)** value is associated, perpendicular to the LC, so the direction of **f(r)** is not determined given **v(r)**

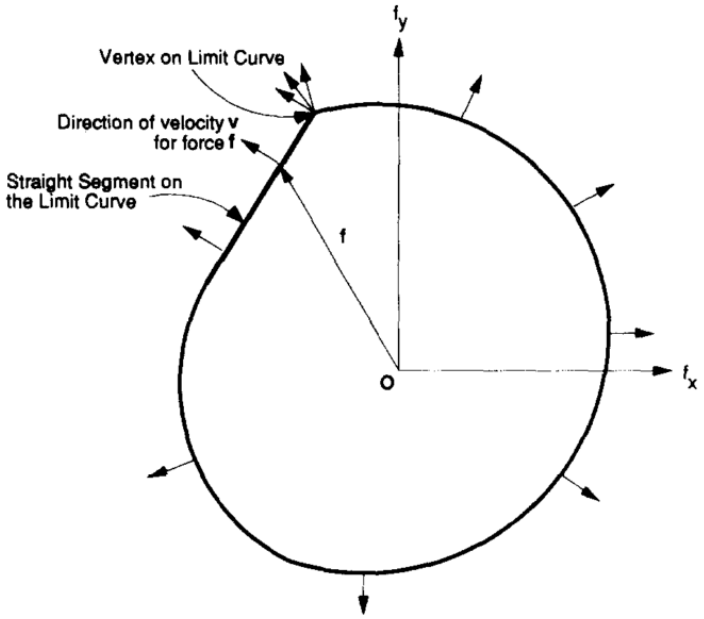

Figure 3: Figure from [12] that shows a general limit curve at a certain contact point. The four possible motion-force relations are represented on it.

If we consider **p** the total frictional load wrench

$$\mathbf{p} = \begin{pmatrix} f_x \\ f_y \\ n_{0z} \end{pmatrix} = \sum_r \begin{pmatrix} f_x(\mathbf{r}) \\ f_y(\mathbf{r}) \\ \mathbf{r} \times \mathbf{f(r)} \end{pmatrix}$$

and **q** the velocity twist

$$\mathbf{q} = \begin{pmatrix} v_{0x} \\ v_{0y} \\ \omega_z \end{pmatrix}$$

then Goyal's study generalizes the concept of single point Limit Curves for force-motion relationship to Limit Surfaces(LS) that relate total wrench and twist for the slider:

Theorem 2.8. Limit Surface:

1. *The set of possible load wrenches* **p** *is contained inside a LS.*
2. *The following relation is held at every contact point:*

$$\forall \mathbf{p}^* \in LS : \ (\mathbf{p} - \mathbf{p}^*) \cdot \mathbf{q} \geq 0 \tag{5}$$

Certain properties are derived for the LS and we are going to list here without proof the ones that apply to our study:

- With isotropic friction, flat regions appear on the limit surface if and only if it has points of finite support (infinite stress).
- With isotropic friction, vertices appear on the limit surface if and only if all support points lie on one straight line segment.

When smooth enough, the limit surface allows to compute the frictional wrench from a non-zero motion twist, but it should be noted that the reciprocal relation can never be obtained (except for the trivial case where the wrench is inside the surface and the twist is zero), but the twist direction can be obtained successfully. As will be discussed in chapter 5, combining the constraints obtained from this relation with other constraints applied on the Pusher-Slider contact, the motion equations for the pusher slider can be derived.

2.3 Contact Pusher-Slider. Friction Cone And Hybridness

The contact between the pusher fingers and the slider sides could be described as in the previous section (by considering the finger a slider and the slider side a support surface) but the nature of the problem allows some simplifications. The fingers we consider never rotate with respect to the slider side, so all the motion is pure translation. Furthermore, we only consider planar pushing (we do not consider operations as tilting the slider), so we are only interested in moving the pusher along one dimension of the slider face (for most purposes we consider the slider to be flat).

A common and useful geometric interpretation of Coulomb's law called the friction cone (according to Mason first constructed by Moseley [23]) is widely used in the robotics community for one dimensional single point contacts. Given a point on a surface interacting with it with total contact force $\mathbf{f}$, we decompose it into its normal f_n and tangential f_t components with respect to the contact surface. Coulomb friction compact formulation states that $f_t \leq \mu f_n$, where μ is the proportionality factor usually called friction coefficient, where the equality holds on dynamic friction. Letting α be the angle between $\mathbf{f}$ and f_n (or, alternatively, the normal of the surface on the point of contact), Coulomb's law is equivalent to:

Theorem 2.9. The set of feasible forces in dry Coulomb contact must satisfy:

$$\alpha := tan^{-1}\frac{f_t}{f_n} \leq \mu$$

So, the set of feasible contact forces lies inside a closed cone with aperture 2μ.

Consider a single point finger pushing a planar slider. There are three possible contact states: The finger can stick to the slider (Sticking State), the slider can slide with a slip to the left relative to the finger (Sliding Left State) or it can slide with slip to the right (Sliding Right State). This section shows that in each state the pusher can apply a different range of forces to the slider. Each of this different range of forces will translate into different ranges of friction reactive wrench, that will ultimately lead to different slider twist ranges.

This multiple system state phenomenon (inherent in Coulomb Contact) is commonly referred as hybridness. Hybridness appears in many manipulation situations different than sticking/sliding ones. For example a gripper can throw the object in the air to re-grasp it, change the number of fingers contacting the object,

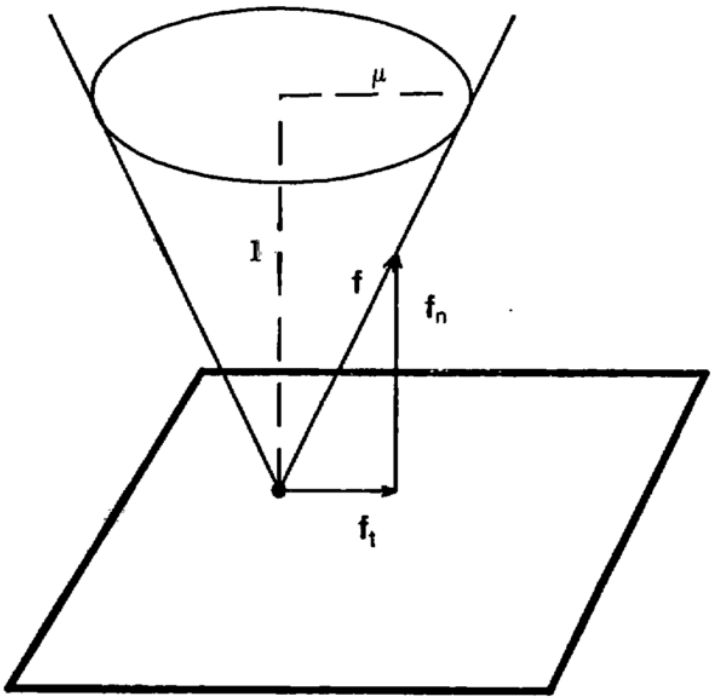

Figure 4: Figure from [20] that shows the friction cone, the geometric interpretation of Coulomb's Law. The set of all vectors following.

change the pusher-slider contact side while pushing a slider of irregular shape, etc. It also appears in other robotic fields as locomotion, where there are always at least two differentiated states (aerial state and ground state) with clearly different dynamics.

In the following chapter, we are going to discuss the hybridness phenomenon in more detail by covering why is it important to take into account, the problems that arise when considering hybridness in planning and control problems and different methods to solve them in each field, that will ultimately lead to the introduction to the FOM method.

3. Hybridness in Manipulation Problems

Through all this section, we will compare the role of hybridness in different aspects of manipulation problems. The principal fields we will distinguish are simulation, trajectory planning and trajectory control. Each of these fields take a different role when studying a general manipulation problem, that we can generally defined as follows:

Definition 3.1. Manipulation Problem:

Given a system conformed by a set of points in a metric space M and a parametrization $c(\mathbf{x})$ (for $\mathbf{x}$ in a certain configuration space C) that uniquely describes the position of all the points of interest in the

metric space, a manipulation problem consists in changing a certain initial configuration $\mathbf{x}_0$ into a final configuration $\mathbf{x}_f$ following appropriate constraints.

Simulation problems consist on accurately reproducing the real physics of a manipulation problem to safely and quickly study its properties. A key property of most simulation problems is that, at every instant of time, we know the configuration of the system until the current time step and the only variables we need to compute are the ones that define the configuration in the next step.

Planning problems consist on finding a trajectory in the configuration state to go from state $\mathbf{x}_0$ to $\mathbf{x}_f$. Most simple planning problems have a state variable that defines the configuration of the object we need to manipulate and a actuator state variable that describes the configuration of the gripper or other tool used to move the object. As opposed to simulation problems, planning problems don't have full information about the state of the object and actuator state variables for each step of the trajectory, and they are usually variables of the problem. Because of this difference, the equations that describe the evolution of the same actuator-object system can have different properties in planning or simulation problems. As an example, the motion equations we derive in section 5.5 are generally non-linear (they are linear with respect to the object state variables but not with respect to the actuator state variables), but they can be expressed as linear equations on the actuator state variables for the first step of the problem, where we know the initial state of the object.

Traditional control designs as PID controllers can successfully control a system by solving only local manipulation problem, in a similar way as simulation. PID only use the notion of the current and past differences between the configuration of the object $\mathbf{x}$ and it's objective value $\mathbf{x}^*$ to devise a control response to stabilize the realized trajectory around the objective one. Such control cannot be used for pushing because of a phenomenon called underactuation.

TODO: Add the concept of WHY Should we care about hybridness here

3.1 Underactuation. Trajectory Optimization

A control problem with first order motion constraints $\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u})$ for some control input $\mathbf{u}$ is underactuated when f is non-exhaustive in the configuration velocities space. In most problems, $\dot{\mathbf{x}}$ is the result of converting a second order Newtonian equation $\ddot{\mathbf{x}} = m \sum F$ to a system of first order ordinary differential equations (ODEs) and underactuation is usually caused by the controller being able to transmit only a limited set of forces and moments to the object.

In stable grasping, the system is fully activated (see remark 3.2) at all times. On the other hand, for a single simple point finger pushing the system is highly underactuated, as the normal force the pusher applies to the slider is non-negative (there cannot be no pulling forces) and the overall contact force lies in the friction cone. As such, the controller must reason only among the forces that can physically be realized at every instant.

In a certain way, it can be considered that underactuation distorts the usual metric of the system. The example in Figure 5 shows a tracking problem where a single point pusher needs to move a square slider to a configuration situated just behind the current pusher location. Any local pushing control action will bring the system away from the objective configuration if we use a tradition 2D Euclidean metric. If the controller takes into account a certain notion of path and future and tries to minimize the overall distance through the considered control trajectory, it can obtain a solution like the one we show.

Remark 3.2. Stable grasping has some some sources of underactuation, but they are linked with the situations were the stable grasp property cannot be held. For example, if an object is too heavy to be lifted

or to be kept in a grasp without slipping. These are examples of underactuation, as the gripper cannot transmit enough force to the object, but they are also examples where stable grasping is not possible.

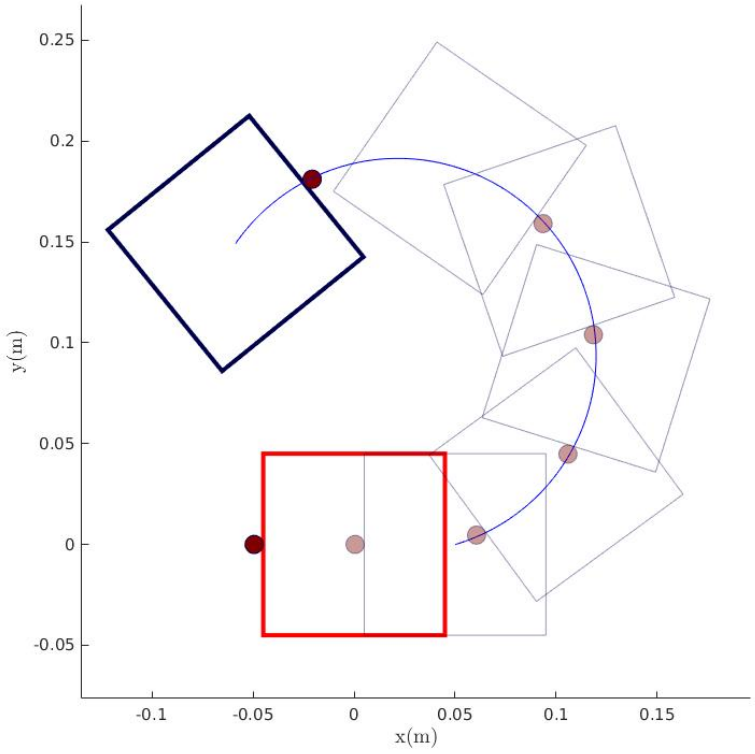

Figure 5: Control trajectory obtained when trying to bring the slider to a configuration behind the current position of the pusher. The red slider describes the objective configuration, the blue line describes the trajectory of the slider center of mass during the control trajectory and the blue sliders represent some of the configurations of the system during the tracking trajectory. The position of the slider when we stopped the simulation is highlighted with thicker and more opaque lines.

A common method to consider future paths in control problems is to solve simplified planning problems at high frequency. The planning problems can usually be expressed as trajectory optimization problems, where the values of the state and control variables are chosen to minimize a cost function along the trajectory, while fulfilling the required motion constraints that represent the kinematics of the real world problem. These optimization problems are usually linearized around the nominal trajectory computed by a planning problem beforehand. This linearization reduces the computational cost of the control problem while ensuring that, in a certain region of attraction, the controlled trajectory converges to the nominal one. It's also common to only consider the control path until a certain finite time horizon (as opposed to planning problems, that usually consider the trajectory until reaching a certain goal).

Once stated the need to consider trajectories in pushing control problems we will now justify the need to consider hybridness in the control problems. This justification is threefold:

- **Optimality:**

- **Robustness:**
- **Convergence:**

In the next sections we will introduce formulation techniques to add the hybridness concept into manipulation problems. In particular, we will discuss the limitations of some of the formulations to solve control problems at enough frequency. This will ultimately lead to Hogan's FOM formulation as an approximate method to solve the finite horizon trajectory optimization problem.

3.2 Complementarity Problems

Given a variable $\mathbf{z}$ and a function on it $\mathbf{g(z)}$, a general complementarity problem consists on finding a value for $\mathbf{z}$ that satisfies:

$$\mathbf{z} \geq 0 \qquad \mathbf{g(z)} \geq 0 \qquad \mathbf{g(z)} \cdot \mathbf{z} = 0$$

This formulation derives from traditional linear programming techniques to encode two mutually exclusive constraints on $\mathbf{z}$ ($\mathbf{z} \geq 0$ and $\mathbf{g(z)} \geq 0$). The linear complementarity problem (LCP) [7] assumes a linear g function and takes the following form:

$$\boldsymbol{\omega} = \mathbf{Rz} + \mathbf{u}$$

$$\boldsymbol{\omega} \geq 0 \qquad \mathbf{z} \geq 0 \qquad \boldsymbol{\omega}^T \cdot \mathbf{z} = 0$$

It was first introduced to predict instantaneous acceleration of rigid bodies in 1996 by Pang and Trinkle [24] and Anitescu and Potra [4], that guaranteed a solution for this formulation on multi-rigid-body Coulomb friction contact and impact problems. This provided a systematic way to formulate hybrid constraints for simulation purposes. Soon after the instantaneous formulation introduction, Steward and Trinkle [26] proposed a time-stepping formulation in the space of impulses and velocities which increased the stability performance of LCP based forward simulation techniques.

Remark 3.3. It should be noted that the LCP formulation of the multi-rigid-body Coulomb friction problems used a polyhedral approximation to the friction cone.

There are fast algorithms to solve LCP problems that allow the manipulation community to use them in simulation, as in Steward and Trinkle's study [26], or other instantaneous-nature problems as the studies of Fazeli et al. to identify friction parameters from observed object motions [10] [9].

In instantaneous Coulomb contact problems the contact dynamics can be expressed in linear discrete form through Steward and Trinkle's formulation,

TODO: talk with nikhil more about it but a key property of their formulation is that they know the value of the velocity and forces prior to contact. but this cannot be applied to contact-driven trajectories. The problem becomes nonlinear (NCP) and an instance of the NCP has to be used to compute the contact state at each point in the discretization of the trajectory.

LCP can not be used as is to solve problems which also contain non-complementarity subject variables that must hold linear constraints. Luckily, these problems, called mixed LCP (MLCP) can also be solved with a generalization of Lemke's algorithm, the key idea used to provide the LCP solution. Nikhil's work on extrinsic dexterity [6] is an example of MLCP usage in planning.

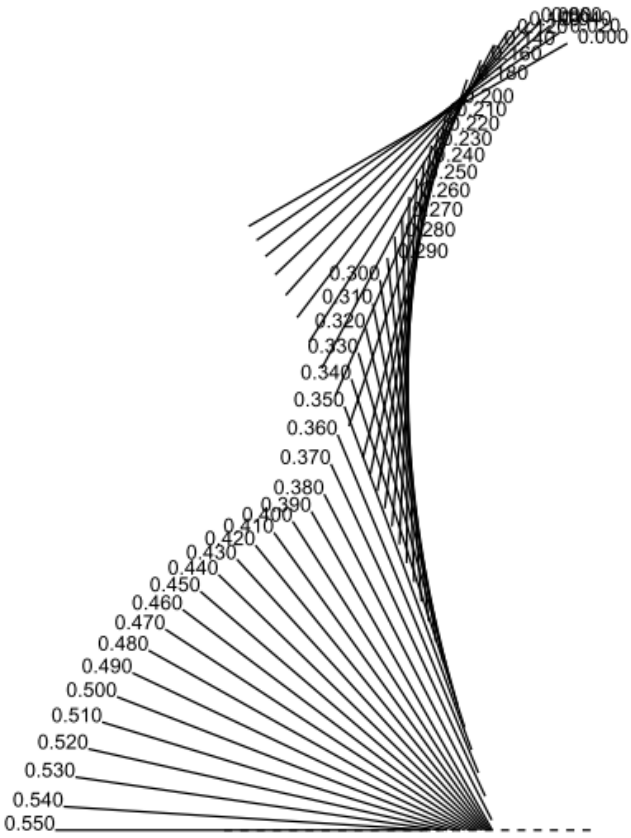

Figure 6: Figure from [26] with the simulation results obtained when emulating a falling and spinning rod. The simulation step was h = 0.0025 and the simulation time at each time instant is shown in seconds on one of the bar ends.

3.3 MPC and MIQP

MLCP is usually too slow to be used in trajectory optimization based control problems. Complementarity constraints are generally non-convex, which makes it difficult to use them in other algorithms than LCP or MLCP problems. To tackle the computing cost of complementarity constraints, model predictive control(MPC) can be used. Given an initial state $\bar{\mathbf{x}}_0$, an MPC problem consists in defining a discrete set of N positions $(\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_N)$ and control inputs $(\bar{\mathbf{u}}_0, ..., \bar{\mathbf{u}}_{N-1})$ to define a trajectory a trajectory. $\bar{\mathbf{x}}_i = \bar{\mathbf{x}}(t_j)$, where $t_j = t_0 + hj$ for a certain time step h. Similarly, $\bar{\mathbf{u}}_i$ stands for the control input necessary to transition from $\bar{\mathbf{x}}_i$ to $\bar{\mathbf{x}}_{i+1}$. Given this notation, MPC optimizes the values of all $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{u}}_i$ with respect to a certain cost function:

$$J(\bar{\mathbf{x}}_0, ..., \bar{\mathbf{x}}_N, \bar{\mathbf{u}}_0, ..., \bar{\mathbf{u}}_{N-1}) = \bar{\mathbf{x}}_N^T \mathbf{Q}_N \bar{\mathbf{x}}_N + \sum_{i=0}^{N-1} \left(\bar{\mathbf{x}}_i^T \mathbf{Q} \bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i^T \mathbf{R} \bar{\mathbf{u}}_i \right) \tag{6}$$

Where terms $\mathbf{Q}$, $\mathbf{Q}_N$, and $\mathbf{R}$ are weights matrices that penalize the error state, final error state, and control input, respectively. MPC problems consider the hybrid state to be constant during state state

transitions. Under this assumption, a hybrid mode can be defined:

Definition 3.4. A **hybrid mode** is a ordered set of hybrid states.

In MPC, an N-length mode completely fixes the hybrid states the system is in along the finite horizon considered trajectory.

Remark 3.5. We use the $\bar{(.)}$ notation because, as will be explained in sections 5.5 and 5.6, we will use the perturbation of the state and control variables with respect to a nominal trajectory and control inputs as the MPC variables.

Remark 3.6. MPC problems are based in the linear quadratic regulators (LQR) controllers. These controllers are used for unconstrained non-hybrid problems described by the motion equation $\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u}$ and use the same cost functions as MPC. LQR are able to give an analytical control law to minimize the cost function provided for infinite horizon problems. Their term $\mathbf{Q}_f$ can be set-up to account for the integral of the cost function from the last point considered until reaching the destination (if the trajectory is finite) or until infinite. If the $\mathbf{Q}_f$ term is chosen to follow the Riccati equation $\mathbf{Q}_f = \mathbf{A}^T \cdot \mathbf{Q}_f \cdot \mathbf{A} - \mathbf{A}^T \cdot \mathbf{Q}_f \cdot \mathbf{B} \cdot (\mathbf{B}^T \cdot \mathbf{Q}_f \cdot \mathbf{B} + \mathbf{R})^{-1} \cdot \mathbf{B}^T \cdot \mathbf{Q}_f \mathbf{A} + \mathbf{Q}$, the finite horizon LQR controllers will behave as the infinite horizon ones. In MPC problems there's no closed expression for this term but there are certain techniques to choose a good value to maximize the final region of attraction as the one exposed in Ya-Feng's et al. study [28], but it is generally based on the LQR technique of solving the Riccati equation . In our model we use a simplification of Ya-Feng's approach which simply computes the solution of the Riccati equation presented above using the motion equation of the last MPC step, that can be expressed as $\dot{\mathbf{x}} = \mathbf{A}_N \cdot \mathbf{x} + \mathbf{B}_N \cdot \mathbf{u}$.

If we consider a manipulation problem with convex dynamics, constraints, and cost function, the overall optimization problem is convex if a hybrid mode that describes the finite horizon is provided. Thus, the global problem consists in selecting the optimal solution from the finite set of optimal solutions for each of the fixed-mode sub-problems.

The number of possible modes of length N is h^N, where h is the number of possible hybrid states. This exponential growth generally makes a naive exploration of all the possible sub-problems too computationally expensive. In order to reduce the algorithmic strain, a mixed integer programming (MIP) strategy is usually adopted. This will be further explained in chapter 5, but the main idea is that the state space is expanded by adding h binary variables per MPC step that define whether the step i of the MPC is in a hybrid state s. The commercial optimizers provide tools to express state the state related constraints with respect to each of this binary variables so that they only apply when their correspondent variable is activated. Then, if possible, the optimizer will use internal MIP techniques as branch and bound and cutting planes [1] to avoid exploring all the possible modes. In our particular case, we will be able to express all the motion and friction constraints as linear inequalities and the objective cost is a quadratic function on the variables. This kind of mixed integer problems are called mixed integer quadratic problems (MIQP) and they have very fast commercial optimizers (as the one we will use, called Gurobi).

The drawback of MIQP is that it doesn't escalate well with the number of possible hybrid states and, even with a small number of states, the optimizers aren't generally fast enough to solve the problem online, as required in control problems. Furthermore, the performance of MIQP depends on the geometry of the problem and it's execution time can change by orders of magnitude at every control iteration, as will be further explained in section 6.3.

Using fast iterations of MPC problems to control a hybrid system is a technique used in other robotics fields, as locomotion, and being able to solve them at a frequency high enough to allow direct implementation into a feedback controller design is currently a problem of interest for the robotics community. A proof of this is that in 2016's Workshop on the Algorithmic Foundations of Robotics (WAFR), held in San

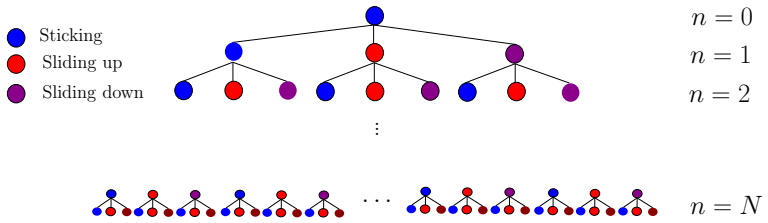

Figure 7: Figure from [14] that represents the exponential growth of the possible modes tree for a single point finger pusher-slider system. The root is a dummy node and, for every other tree level i, each node represents a possible hybrid state for the $i_t h$ state transition. Each possible mode can be represented by a path from the root to one of the tree leaves

Francisco last December, Tedrake presented it as one of the current algorithmic open problems in robotics control problems. TODO: Cite

3.4 Family of Modes

Responding to the need of obtaining fast solutions to MPC-based control problems Hogan and Rodriguez [14] proposed an heuristic based procedure to approximately solve them called Family of Modes (FOM). In their work, they assumed a finite horizon MPC problem as the one described in the previous section. At every control step the heuristic method explores the optimization problem over only a small set of all the possible combinations of hybrid states to decide the instantaneous controller policy.

The approximate MPC is characterized by a family of modes, which is defined as the set of modes to consider. If the MPC family contains K modes, the MPC will only solve the K optimization sub-problems resulting when enforcing each mode's hybrid state schedule. The optimal solution is approximated by the best solution between the K considered. This solution will consist of the values of $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{u}}_i$ for the optimal control trajectory. The controller will then keep executing the first control policy $\mathbf{u}_1$ and recomputing the approximate MPC problem frequently to take into account new information as the finite horizon advances in time as well as to tackle external interference with the system and errors in the dynamic model.

In his study, Hogan applied FOM to a system consisting on a single finger point pusher and a square slider and he showed that his method was capable of properly tracking straight line trajectories (and composition of straight lines) under external perturbations by using only three modes. The three modes were selected by hand using intuition on the problem structure. They consisted on exploring the three possible hybrid states (using the notation of section 2.3 they were Sticking, Sliding Right, and Sliding Left) on the first step of the MPC and constraining the system to be in a Sticking state for the rest of the steps.

In this simple case it is possible to derive a good set of modes intuitively, but for more complex and higher dimensional problems, this may not be possible. Furthermore, adding more modes to consider into this heuristic doesn't generally translate into better results and, in some cases, it may lead to results far worse than the original one. For some cases, adding new modes to a family that was able to track a line providing asymptotic convergence could even translate into losing this convergence.

We believe the FOM approach shows promising results and may be able to partially address the current needs of manipulation research to solve the MPC based hybrid control problem. At the same time, we consider necessary to show it's generality and robustness as well as address some of its limitations for it to be adopted in the robotics community.

In the next section we introduce key concepts on sequential properties of heuristic based methods, that will be useful to describe some of the FOM limitations described above. In chapter 4 we will revisit the main objectives of our study using the concepts presented in previous sections.

3.5 Sequential Properties

On the late 90's, Bertsekas et al. presented a work on the sequential properties of heuristic algorithms [5]. Two important definitions introduced in their study are sequential consistency and sequential improvement. They also introduce the concept of a roll-out algorithm, which holds a close resemblance to the FOM approach to solve MPC problems. At the end of the section we will further discuss this, but we first need to introduce the main definitions and conclusions of the sequential properties study. We will use the same introductory example used in their study because it is similar to the problem we try to solve. The main differences will be mentioned at the end of the section.

Let us consider a graph search problem on a directed graph with node set N and arc set A. For simplicity, an initial node s is defined. For consistency, we will keep the formulation presented in Bertsekas' study, that groups all the search costs in a subset of terminal (with no outgoing arcs) nodes $\bar{N}$ called destinations. In this formulation, each path from the origin to a destination node $(s, i_1, i_2, ..., i_n, \bar{i})$ (see remark 3.7) has an associated cost $g(\bar{i})$ solely defined by the destination node. We want to find a directed path that starts at the origin s, ends at one of the destination nodes $i \in \bar{N}$, and is such that the cost g(i) is minimized.

Remark 3.7. In order for a sequence $(i_1, i_2, ..., i_n)$ to uniquely define the path created by the composition of the arcs $(i_1, i_2), (i_2, i_3), ..., (i_{n-1}, i_n)$ we need to assume that given an ordered pair of nodes (i, j), there is at most one arc (i, j) with start node i and end node j. This assumption can always be safely made.

Remark 3.8. We allow the node and arc sets, N and A, to contain an infinite number of elements. We require, however, that the number of destination nodes be finite.

The search problem can then be formulated as a DP problem with the nodes corresponding to the states of the DP and the arcs corresponding to the possible state transitions, and the destination nodes corresponding to the terminal states. An heuristic H is then defined as a path construction algorithm that construct a path from a non-destination node to a destination one (remark 3.9) and $H(i)$ is defined for any node $i \in N$ as the cost of the destination node of the path constructed by H:

$$H(i) = g(\bar{i})$$

Remark 3.9. Implicit in this assumption is that for every non-destination node, there exists at least one path starting at that node and ending at some destination node.

Having introduced this definition of heuristic in this search problem, the concept of roll-out algorithm, first used by Tesauro and Galperin (TODO: Citation) in 1996, is introduced. The formal definition of roll-out algorithm escapes the scope of this study, but an intuitive definition would be:

Definition 3.10. Roll-out Algorithm:

Given a starting node s and a heuristic H, we define the roll-out algorithm based on H (called RH) as following:

1. We consider all downstream neighbours i (with $|i| = n$) of s and we apply H at each possible i.
2. We consider the n paths $(i, i_1, ..., \bar{i})$ obtained from the previous step, each of which has an associated cost $H(i)$.

3. We compute the node i^* which minimizes $H(i)$ and we move to the neighbour i_1^*.

4. The algorithm is then iteratively developed by considering i_1^* the new start node s and repeating the procedure from step 1 until s is a terminal node.

The nature of the roll-out algorithm and the FOM approach are similar in the sense that use simpler base heuristics that would not give satisfying solutions by themselves and keep applying them at every new step, exploring new paths that were not considered when the heuristics were applied to the root node. This study provided some interesting properties that related the heuristic H with its roll-out RH. Most of them were properties that guaranteed that RH would eventually stop (a problem we don't have in our MPC approach, because $|N|$ and $|A|$ are finite) or variations of the roll-out technique that we will discuss later. In order to properly explain them we need to first introduce two main concepts that helped us to understand some of the limitations of the current FOM formulation and that will be discussed on section 6.4.

Definition 3.11. Sequential Consistency:

The heuristic H is said to be sequentially consistent if for every node i, whenever H generates the path $(i, i_1, ..., i_m, \bar{i})$ starting at i, it also generates the path $(i_1, ..., i_m, \bar{i})$ starting at the node i_1.

Definition 3.12. Sequential Improvement:

Suppose that algorithm H generates, starting at each node $i \in N$, a path $(i, i_1, ..., i_m, \bar{i})$ with the property $H(i) \geq H(i_1)$. Then the heuristic H is said to be sequentially improving.

We are going to borrow two of Bertsekas' examples (associated with Figure 8) to illustrate this two concepts.

Example 3.13. One dimensional walk with a sequentially consistent heuristic:

Consider a person who walks on a straight line in a temporal and spatial discrete world. During each time step, it can move a fixed length unit to the right or to the left (it cannot stay still). At time 0 this person is at position 0 and it walks for a fixed number of time steps N before ending up in a position $\bar{i} \in [-N, N]$. Each terminal point has an associated cost $g(\bar{i})$ and the objective is to minimize the cost of the terminal position. Figure 8 shows a graph representation of the analogue graph search problem, where each node is defined by a state (k, m) where k is the number of steps taken so far and m is the current position. Each node (k, m) has two outgoing arcs: to $(k+1, m-1)$ for the left step and to $(k+1, m+1)$ for the right one.

Let H_1 be an heuristic defined as follows: If H_1 applied at node (k, m), it takes $N-k$ successive steps to the right, terminating at the node $(N, m+N-k)$. Note that H_1 clearly is sequentially consistent.

The algorithm RH_1 applied at node (k, m) uses the roll-out algorithm described above and obtains the path highlighted in Figure 8. In Bertsekas' study it is further proven that the obtained result is always a local minimum.

Example 3.14. Consider the set-up of example 3.13, and let H_2 be defined as follows: when applied at node (k, m), H_2 compares the cost $g(m+N-k)$ (corresponding to taking all of the remaining $N-k$ steps to the right) and the cost $g(m-N+k)$ (corresponding to taking all of the remaining $N-k$ steps to the left). If the first is lower, it takes one step to the right. Otherwise, it takes one step to the left. In Bertsekas' work it is mentioned that H_2 has no sequential consistency propriety, but has instead sequential improvement. It is also shown that RH_2 obtains the global minimum.

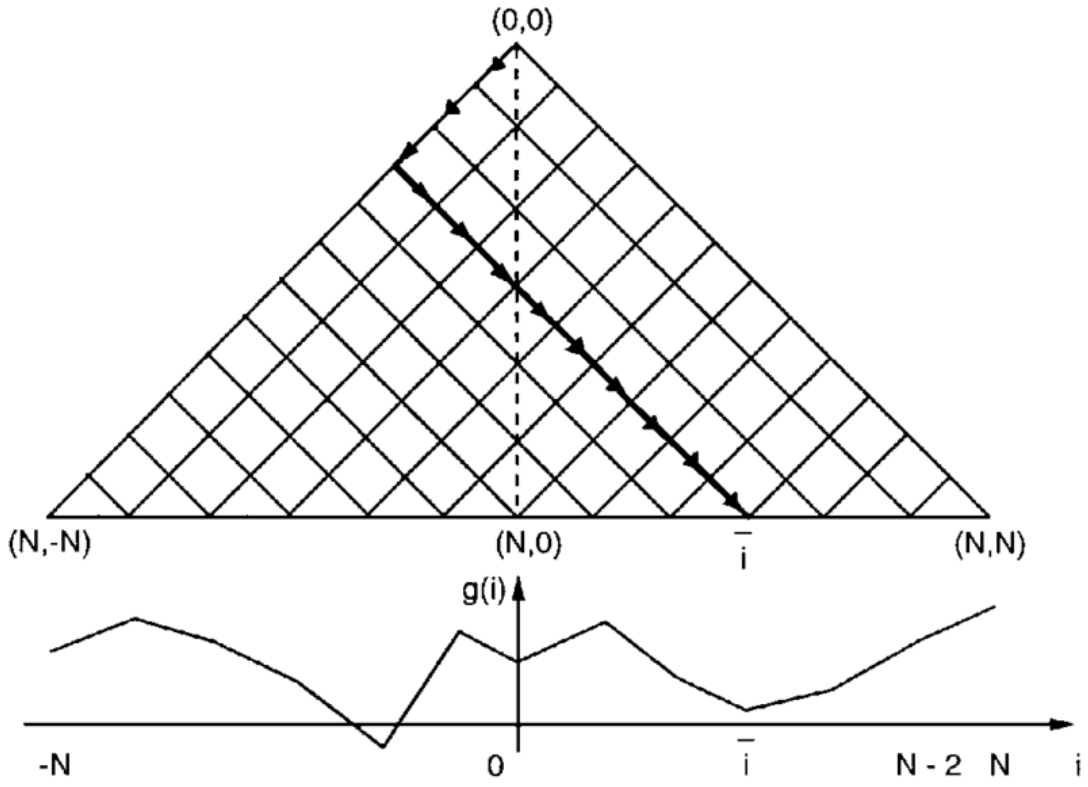

Figure 8: Excerpt from Bertsekas' study. It shows the graph associated with a one dimensional walk problem. The highlighted path corresponds to the result obtained by applying both the roll-out algorithm RH_1 from example 3.13. A plot of the cost function g(i) is shown on the lower part of the picture.

Using these definitions, Bertsekas et al. derive some relations between the base heuristics and their associate roll-out procedure. In section 6.4 we propose a method to guarantee the sequential improvement of the FOM method when used with a full horizon instead of a finite one. Our method was inspired in a modification of the traditional roll-out technique proposed by Bertsekas called optimal roll-out.

Consider that at a given roll-out step, the heuristic H generates the optimal path $(i_0, i_1, ..., i_m, \bar{i})$. Then the fortified roll-out algorithm stores the maximal suffix $S_0 = (i_1, ..., i_m, \bar{i}$ and, in the next roll-out iteration, it will consider the paths generated by H together with the path defined by S_0. If the optimal path given by H is not better than the one provided by S_0, the algorithm executes the next step of S_0 and stores the next maximal suffix $S_1 = (i_2, ..., i_m, \bar{i}$. If the path provided by H is better, its first step is executed and the algorithm stores its maximal suffix. This procedure is followed until reaching one of the terminal nodes.

The main idea behind this modified roll-out technique is to simply keep comparing all the possible paths at a certain step with the best path from the previous step. This way the sequential improvement of the algorithm is clearly guaranteed.

4. Goal Revisited

To solve MPC problems fast enough to implement them online in a control problem is an open problem of interest for the robotics manipulation community, as explained in section 3.3. We consider that Hogan's work in the FOM technique is a good way to provide an heuristic-based approximate solution to this problem. The main objective of this study is to bring this method close to the research community by:

1. **Improving the properties of FOM:** In section 6.4 we will discuss some of it's sequential weaknesses and the solutions we provided and in section 7.3 we will present a new FOM-based method to track complex trajectories.

2. **Comparing it to other techniques:** In section 6.3 we will discuss some of the difficulties we found while trying to compare it to MIQP and we present a modified version of the method to address this problem.

3. **Providing a systematic set-up:** In chapter 7 we will present a systematic protocol with an stochastic approach to choose the modes of the family.

4. **Showing the system scalability and robustness:** We want to track more complex trajectories, more complex systems, and use models of a simple system to solve a problem that uses a more complex one. This study contains results for some of these studies, and they will be discussed through chapters 6 and 7. We will also present the studies and experiments we are currently developing for ISRR in chapter 9.

In order to achieve this goals, we provided quantitative tools to evaluate the obtained trajectories that will be presented in section 6.1 and we implemented a general simulation interface (presented in chapter 6) that allowed us to easily formulate and solve general trajectories and hybrid systems.

The formulation used in our study will be presented in the next chapter, together with all its required theoretic concepts.

5. Pusher-Slider Formulation

In this section we are going to show all the necessary steps to formulate the FOM problem for the case of a rectangular pusher slider-system with a single finger point pusher.

5.1 Nomenclature Summary

Property	Symbol	Value
Coeff. friction (Pusher-Slider)	μ_p	0.3
Coeff. friction (Slider-Ground)	μ_g	0.35
Mass of the slider (Kg)	m	1.05
Length of the slider (m)	a	0.09
Width of the slider (m)	b	0.09

Table 1: Properties of the slider and the support surface

Property	Symbol
Length of the finite horizon	N
Time step	h
State cost matrix	$\mathbf{Q}$
Control input cost matrix	$\mathbf{R}$
Final state cost matrix	$\mathbf{Q}_f$

Table 2: MPC variable summary

Property	Symbol
Position of the slider's CM	$(x, y)^T$
Slider orientation	θ
Slider inertial frame	F
Pusher position in F	$(p_x, p_y)^T$
Pusher velocity in F	$(v_x, v_y)^T$
Trajectory state	$\mathbf{x} = (x, y, \theta, p_y)^T$
Control input	$\mathbf{u} = (v_x, v_y)^T$

Table 3: Pusher-Slider system variable summary

To derive the motion equations and define the MPC state and controller variables, we are going to use the notation used in [14]. We will first define an inertial reference frame F centred on the slider center of mass that will move and with it. The pose $\mathbf{q}$ of the slider's center of mass will be defined in a fixed world frame $\mathbf{q} = (x, y, \theta)^T$ and the pose $\mathbf{p}$ of the pusher will be defined relative to the slider center of mass $\mathbf{p} = (p_x, p_y)^T$ as shown in Figure 9. The state variable used to define a trajectory is $\mathbf{x} = (x, y, \theta, p_y)^T$ and the control input is the velocity of the pusher defined in the center of mass frame $\mathbf{u} = \mathbf{v}_p = (v_x, v_y)^T$.

When talking about hybrid states and modes, we will usually use the following naming convention:

- S: Sticking State.
- R: Sliding Right State.

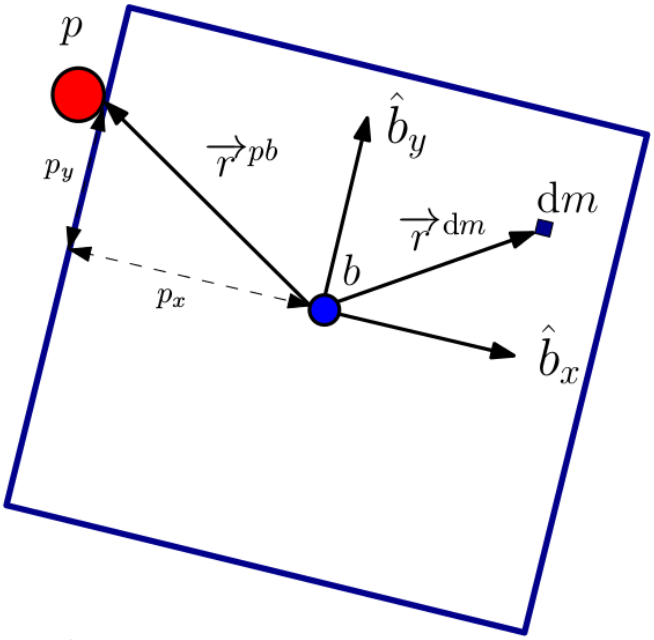

Figure 9: Figure from [14] that shows the inertial frame used to define some of the state variables

- L: Sliding Left State.
- $\mathbf{S}^N$: Mode defined by N S states $(S, \overset{N}{\cdots}, S)$.
- $\mathbf{R}_i^N$: Mode defined by $N-1$ states S and one state R in position i. For example, $\mathbf{R}_1^3 = (R, S, S)$.
- $\mathbf{L}_i^N$: Analogue to $\mathbf{R}_i^N$. Mode defined by $N-1$ states S and one state L in position i.

5.2 Quasistatic Pushing Formulation

Quasistatic analysis consists in assuming balance among contact wrenches, gravitation and other applied forces, while neglecting inertial wrenches. This analysis is often taken in manipulation problems, where the trajectory tracking speed changes don't incur into high inertial forces.

The slider of our pusher-slider system with a single finger point pusher is affected by two wrenches: the wrench the pusher applies on the slider ($\mathbf{f}_p$) and the wrench the support surface applies to it ($\mathbf{f}_g$). Under the quasistatic assumption $\mathbf{f}_p = -\mathbf{f}_g$.

This assumption will reduce the dimension of the state variables in the motion equations derived in section 5.5 and will allow as to ignore considering a separation hybrid state in our hybrid model of the pusher slider system (if inertial forces were non-negligible the slider could separate away from the pusher while tracking complex trajectories with points of high curvatures). Furthermore, combining the quasistatic assumption with the limit surface of the slider-ground contact, we can derive velocity based motion equations and friction constraints, as will be shown in section 5.5. The robot arms we had available for this study are velocity-controlled, so this formulation simplifies the experimental implementation of the controller.

5.3 Ellipsoidal Approximation Of The Limit Surface

Given a pressure distribution, the associated surface distribution can be numerically computed. Unluckily, the pressure distribution is usually not known exactly and changes over time. For our this study we will assume that the slider and the supporting surface are completely flat, providing a uniform pressure distribution, so $f_n = \frac{gm}{A}$, where f_n is the magnitude of the normal support force the ground exerts on the slider and m and A are the slider mass and area, respectively. Furthermore, we will use an approximation to the actual limit surface proposed by Lee and Cutkosky on 1991 [17] based on a study made on 1988 by Howe, Kao, and Cutkosky that showed the suitability of using an elliptic approximation for dry Coulomb friction [15]. We will add further simplifications to their ellipsoid derivation. For a single contact point, their derivation can be simplified, as the limit surface will already be oriented in the ellipsoid major axis. Assuming uniform pressure distribution also simplifies their computation of the maximum momentum around the instantaneous center of rotation. The final procedure to obtain the LS ellipsoid is as follows:

1. Compute the maximum momentum value. This occurs when the object is in pure rotation, so we can use equation 4 considering $\mathbf{r}_{IC}$ to be the center of mass (CM) of the slider:

$$n_{max} = \left| -\mu \, sign(\dot{\theta}) \int_F \mathbf{r} \cdot \frac{\mathbf{r} - \mathbf{r}_{CM}}{|\mathbf{r} - \mathbf{r}_{CM}|} p(\mathbf{r}) dA \right| = \mu \frac{gm}{A} \int_F ||\bar{\mathbf{r}}|| dA \tag{7}$$

   where $\bar{\mathbf{r}}$ is the vector from $\mathbf{r}_{CM}$ to $\mathbf{r}$. This gives us the value of the moment semi-axis (the maximum and minimum values of m in the ellipse are going to be $\pm n_{max}$).

2. Similarly, to obtain the force semi-axes, we need the maximum force in the x and y directions, which is going to be $f_{max} = f_n$.

3. The final expression for the ellipsoid is:

$$\left(\frac{f_x}{f_{max}}\right)^2 + \left(\frac{f_y}{f_{max}}\right)^2 + \left(\frac{n}{n_{max}}\right)^2 = 1 \tag{8}$$

5.4 Motion Cone

In Hogan's study, a velocity controlled pusher was used. For this reason the motion equations for the MPC problem were formulated in the velocity space. For the case of a single point finger pushing, we can obtain an expression for the friction cone in the velocity space called motion cone (MC) [20] that will be useful to formulate the motion equations and the friction constraints.

As opposed to friction cone, the motion cone is not symmetric with respect to the normal of the contact surface at the point of contact. The cone can be determined by using two angles $\gamma_t(\mathbf{p})$ and $\gamma_b(\mathbf{p})$. Using the ellipsoidal approximation of the LS, the expression of the motion cone for flat sliders can be given as:

$$\gamma_t = \frac{\mu_p c^2 - p_x p_y + \mu_p p_x^2}{c^2 + p_y^2 - \mu_p p_x p_y} \tag{9}$$

$$\gamma_b = \frac{-\mu_p c^2 - p_x p_y - \mu_p p_x^2}{c^2 + p_y^2 + \mu_p p_x p_y} \tag{10}$$

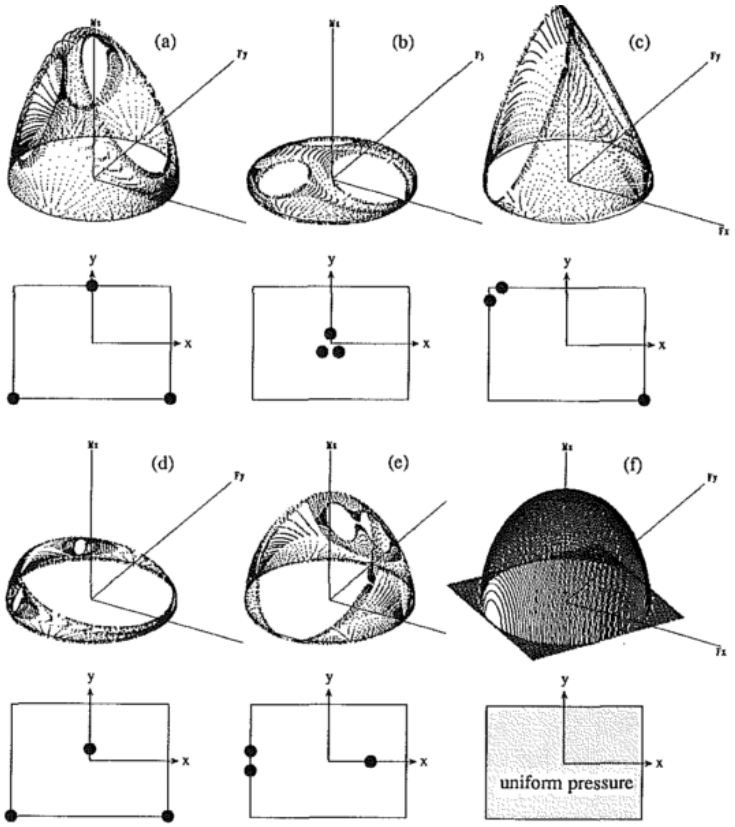

Figure 10: Figure from [17] that shows the experimental limit surface for different contact assumptions (that result in different pressure distributions). For the uniform pressure distribution the ellipsoidal approximation describes accurately the LS

where p_x and p_y have been defined in section 5.1 and $c = \frac{f_{max}}{n_{max}}$.

We will define $\gamma := \frac{v_y}{v_x}$. If the velocity vector lies inside the motion cone, the pusher will stick. If $\gamma > \gamma_t$ the slider will slip to the right with respect to the pusher. Finally, if $\gamma < \gamma_b$ the slider will slip to the left with respect to the pusher. In summary the friction constraints can be expressed for each hybrid state as:

$$\textbf{Sticking:} \quad \mathbf{u} \in MC : \begin{cases} v_y \leq \gamma_t v_x \\ v_y \geq \gamma_b v_x \end{cases} \tag{11}$$

$$\textbf{Sliding Right:} \quad \mathbf{u} > MC : v_y > \gamma_t v_x \tag{12}$$

$$\textbf{Sliding Left:} \quad \mathbf{u} < MC : v_y < \gamma_b v_x \tag{13}$$

TODO: Decide what to do with Maria's piece of advice on these equations.

In our single finger point pusher system the three possible hybrid states of the system will be these. The quasistatic assumption guarantees that the slider doesn't move away from the pusher by virtual inertial

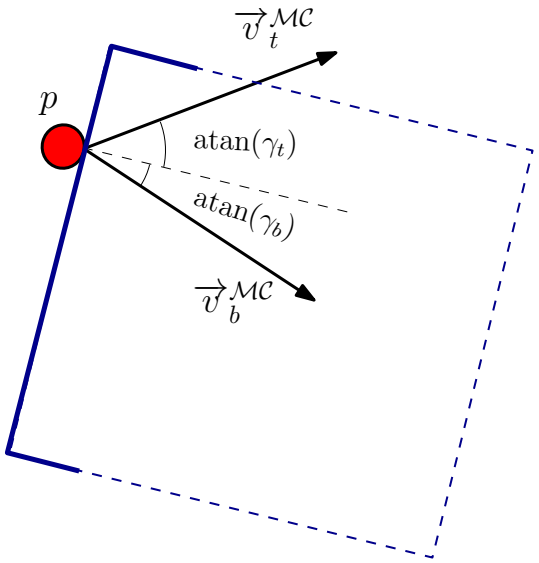

Figure 11: Figure from [14] with a representation of the motion cone

forces and we will impose constraints on the pusher to prevent it to move away from the slider: the normal velocity of the pusher with respect to the contact surface will be non-negative and the tangential displacement from the contact surface center (p_y) will be constrained to prevent the pusher from slipping through the slider sides.

5.5 Motion Equations

The motion equations are adapted from the ones used in Hogan's work. They were originally inspired in a pusher-slider system study by Lynch et al. [18] and can be expressed as follows:

(TODO: Should I add the derivation from Lynch?)

$$\dot{\mathbf{x}} = \begin{cases} f_1(\mathbf{x}, \mathbf{u}) \text{ if } \mathbf{u} \in \text{MC} \\ f_2(\mathbf{x}, \mathbf{u}) \text{ if } \mathbf{u} > \text{MC} \\ f_3(\mathbf{x}, \mathbf{u}) \text{ if } \mathbf{u} < \text{MC} \end{cases} \tag{14}$$

$$f_i(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \mathbf{CQP}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \end{pmatrix} \mathbf{u} := \hat{\mathbf{B}}_j(\mathbf{x}) \cdot \mathbf{u} \tag{15}$$

where $\mathbf{C} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$ and $\mathbf{Q} = \frac{1}{c^2 + p_x^2 + p_y^2} \begin{pmatrix} c^2 + p_x^2 & p_x p_y \\ p_x p_y & c^2 + p_y^2 \end{pmatrix}$.

$\mathbf{P}_i$, $\mathbf{b}_i$ and $\mathbf{c}_i$ are mode-dependant matrices and take the following values:

$$\mathbf{b}_1 = \frac{1}{c^2 + p_x^2 + p_y^2} \begin{pmatrix} -p_y & p_x \end{pmatrix}, \ \mathbf{b}_2 = \frac{1}{c^2 + p_x^2 + p_y^2} \begin{pmatrix} -p_y + \gamma_t p_x & 0 \end{pmatrix}, \ \mathbf{b}_3 = \frac{1}{c^2 + p_x^2 + p_y^2} \begin{pmatrix} -p_y + \gamma_b p_x & 0 \end{pmatrix}$$

$$\mathbf{c}_1 = \begin{pmatrix} 0 & 0 \end{pmatrix}, \qquad \mathbf{c}_2 = \begin{pmatrix} -\gamma_t & 1 \end{pmatrix}, \qquad \mathbf{c}_3 = \begin{pmatrix} -\gamma_b & 1 \end{pmatrix}$$

$$\mathbf{P}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{P}_2 = \begin{pmatrix} 1 & 0 \\ \gamma_t & 0 \end{pmatrix}, \qquad \mathbf{P}_3 = \begin{pmatrix} 1 & 0 \\ \gamma_b & 0 \end{pmatrix}$$

where i = 1, 2, 3 correspond to the states of Sticking, Sliding Right and Sliding Left respectively. For simplicity, we do not consider a state where the pusher is separated from the slider for the controller modelling. In the controller we impose constraints that forbid the pusher to move outside of the slider side borders and the under the quasistatic assumption, the slider will not separate from the pusher by it inertia.

On the other hand the possibility of separation is included in the simulator, in order to represent the experimental set-up more accurately and to detect possible errors in the controller design.

The motion equations are going to be used to define constraints in the MPC problem and, in turn, the MPC formulation is going to be fed into a commercial optimizer to obtain the optimal values of $\mathbf{x}$ and $\mathbf{u}$ that fulfil all the constraints. The fact that the motion equations are nonlinear would require using a nonlinear optimizer, which would represent too much computational cost to solve the problem online at every control iteration. In order to obtain a more tractable formulation the motion equations will be linearized around a nominal trajectory $\mathbf{x}^*(t)$ and control input $\mathbf{u}^*(t)$. We will use $(.)^*$ to denote elements of the nominal problem and $\bar{(.)}$ to denote perturbations around the nominal value. Using this notation, the linearization of the motion equation 14 is:

$$\dot{\bar{\mathbf{x}}} = \mathbf{A}_i(\mathbf{x}^*, \mathbf{u}^*) \cdot \bar{\mathbf{x}} + \mathbf{B}_i(\mathbf{x}^*, \mathbf{u}^*) \cdot \bar{\mathbf{u}} + f_i(\mathbf{x}^*, \mathbf{u}^*) - \dot{\mathbf{x}}^* \tag{16}$$

$$\mathbf{A}_i(\mathbf{x'}, \mathbf{u'}) = \left.\frac{\partial f_i(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x'}, \mathbf{u'}}, \; \mathbf{B}_i(\mathbf{x'}, \mathbf{u'}) = \left.\frac{\partial f_i(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x'}, \mathbf{u'}} = \hat{\mathbf{B}}_i(\mathbf{x'}) \text{ ,with } \bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^* \text{ and } \bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$$

It should be noted that equation 16 is different from the linearized motion equation presented in Hogan's work. This fact will be discussed further in section 6.2 but the main idea is that the term $f_i(\mathbf{x}^*, \mathbf{u}^*) - \dot{\mathbf{x}}^*$ cannot generally be cancelled as in other traditional control problems because $f_i(\mathbf{x}^*, \mathbf{u}^*)$ will not describe the instantaneous motion of the nominal trajectory if the nominal trajectory is in a hybrid state different from the one defined by i.

Similarly to the linearization of the motion equation, the friction constraint equations from 11 to 13 can also be linearized with respect to $\bar{\mathbf{x}}$. For consistency with the variables used for the motion equations ($\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$) v_x and v_y are expressed as $v_x^* + \bar{v}_x$ and $v_y^* + \bar{v}_y$ respectively:

$$\textbf{Sticking:} \quad \begin{cases} (v_y^* + \bar{v}_y) \leq (\gamma_t(\mathbf{x}^*) + \mathbf{C}_t(\mathbf{x}^*) \cdot \bar{\mathbf{x}})(v_x^* + \bar{v}_x) \\ (v_y^* + \bar{v}_y) \geq (\gamma_b(\mathbf{x}^*) + \mathbf{C}_b(\mathbf{x}^*) \cdot \bar{\mathbf{x}})(v_x^* + \bar{v}_x) \end{cases} \tag{17}$$

$$\textbf{Sliding Right:} \quad (v_y^* + \bar{v}_y) > (\gamma_t(\mathbf{x}^*) + \mathbf{C}_t(\mathbf{x}^*) \cdot \bar{\mathbf{x}})(v_x^* + \bar{v}_x) \tag{18}$$

$$\textbf{Sliding Left:} \quad (v_y^* + \bar{v}_y) < (\gamma_b(\mathbf{x}^*) + \mathbf{C}_b(\mathbf{x}^*) \cdot \bar{\mathbf{x}})(v_x^* + \bar{v}_x) \tag{19}$$

where $\mathbf{C}_t(\mathbf{x'}) = \left.\frac{\partial \gamma_t}{\partial \bar{\mathbf{x}}}\right|_{\mathbf{x'}}$ and $\mathbf{C}_b(\mathbf{x'}) = \left.\frac{\partial \gamma_b}{\partial \bar{\mathbf{x}}}\right|_{\mathbf{x'}}$

This notation allows us to write all the linearized motion cone constraints in a condensed form suitable for a linear programming optimizer, as equation 16:

$$\mathbf{E}_i(\mathbf{x}^*, \mathbf{u}^*) \cdot \bar{\mathbf{x}} + \mathbf{D}_i(\mathbf{x}^*) \cdot \bar{\mathbf{u}} \leq g_i(\mathbf{x}^*, \mathbf{u}^*) \tag{20}$$

$$\mathbf{E}_1(\mathbf{x'},\mathbf{u'}) = v'_y \begin{pmatrix} -\mathbf{C}_t(\mathbf{x'}) \\ \mathbf{C}_b(\mathbf{x'}) \end{pmatrix}, \qquad \mathbf{E}_2(\mathbf{x'},\mathbf{u'}) = v'_y \mathbf{C}_t(\mathbf{x'}), \qquad \mathbf{E}_3(\mathbf{x'},\mathbf{u'}) = -v'_y \mathbf{C}_b(\mathbf{x'})$$

$$\mathbf{D}_1(\mathbf{x'}) = \begin{pmatrix} -\gamma_t(\mathbf{x'}) & 1 \\ \gamma_b(\mathbf{x'}) & -1 \end{pmatrix}, \qquad \mathbf{D}_2(\mathbf{x'}) = \begin{pmatrix} -\gamma_t(\mathbf{x'}) & -1 \end{pmatrix}, \qquad \mathbf{D}_3(\mathbf{x'}) = \begin{pmatrix} -\gamma_b(\mathbf{x'}) & 1 \end{pmatrix}$$

$$\mathbf{g}_1(\mathbf{x'},\mathbf{u'}) = \begin{pmatrix} -v'_y + \gamma_t(\mathbf{x'})v'_x \\ v'_y - \gamma_b(\mathbf{x'})v'_x \end{pmatrix}, \qquad \mathbf{g}_2(\mathbf{x'},\mathbf{u'}) = v'_y - \gamma_t(\mathbf{x'})v'_x - \epsilon, \qquad \mathbf{g}_3(\mathbf{x'},\mathbf{u'}) = -v'_y + \gamma_b(\mathbf{x'})v'_x - \epsilon$$

Remark 5.1. ϵ is just a small positive scalar used to express strict inequalities, as the optimizer we used only supported non-strict ones.

5.6 MPC And MIQP Formulation

As explained in section 3.3, each MPC mode fixed sub-problem will compute the optimal values for the state and control variables for a discrete finite horizon of N steps, a time step h, and a given initial position $\bar{\mathbf{x}}_0$ to minimize the cost equation provided in equation 6, subject to motion and cone constraints. We will use the perturbation dynamics, so the state and control variables will be $(\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_N)$ and $(\bar{\mathbf{u}}_0, ..., \bar{\mathbf{u}}_{N-1})$; and the initial state will be $\mathbf{x}_0$. As MPC considers a discrete framework, we need to discretize the motion equations from previous section. After a first order discretization, the MPC constraints at step j are:

$$\text{if } j = 0 : \begin{cases} \bar{\mathbf{x}}_1 = \bar{\mathbf{x}}_0 + h\mathbf{B}_{i0}(\mathbf{x}_0) \cdot \bar{\mathbf{u}}_0 + h\mathbf{f}_{i0} + \mathbf{x}^*_0 - \mathbf{x}^*_1 \\ \mathbf{D}_{i0} \cdot \bar{\mathbf{u}}_0 \leq \mathbf{g}_{i0} \end{cases} \tag{21}$$

$$\text{if } j > 0 : \begin{cases} \bar{\mathbf{x}}_{j+1} = (\mathbf{I} + h\mathbf{A}_{ij}) \cdot \bar{\mathbf{x}}_j + h\mathbf{B}_{ij} \cdot \bar{\mathbf{u}}_j + h\mathbf{f}_{ij} + \mathbf{x}^*_j - \mathbf{x}^*_{j+1} \\ \mathbf{E}_{ij} \cdot \bar{\mathbf{x}}_j + \mathbf{D}_{ij} \cdot \bar{\mathbf{u}}_j \leq \mathbf{g}_{ij} \end{cases} \tag{22}$$

where $\mathbf{A}_{ij} = \mathbf{A}_i(\mathbf{x}^*_j, \mathbf{u}^*_j)$, and $\mathbf{E}_{ij} = \mathbf{E}_i(\mathbf{x}^*_j, \mathbf{u}^*_j)$, using the notation from section 5.5. $\mathbf{f}_{ij}$ is defined differently depending on the iteration step:

$$\mathbf{f}_{ij} = \begin{cases} \mathbf{f}_i(\mathbf{x}_j, \mathbf{u}^*_j), & (\text{if j} = 0) \\ \mathbf{f}_i(\mathbf{x}^*_j, \mathbf{u}^*_j), & (\text{if j} > 0) \end{cases}$$

$\mathbf{B}_{ij}$, $\mathbf{D}_{ij}$, $\mathbf{E}_{ij}$, and $\mathbf{g}_{ij}$ are defined in an analogue way, evaluating their section 5.5 counterparts on $\mathbf{x}_j$ for the first step and on $\mathbf{x}^*_j$ for the following ones.

Remark 5.2. For the case j = 0, $\bar{\mathbf{x}}_0$ is known value, so the first order discrete version of equation 14 becomes:

$$\mathbf{x}_1 = \bar{\mathbf{x}}_1 + \mathbf{x}^*_1 = h\mathbf{f}_i(\mathbf{x}_0, \mathbf{u}_0) + \mathbf{x}_0 = h\hat{\mathbf{B}}_i(\mathbf{x}_0)(\bar{\mathbf{u}}_0 + \mathbf{u}^*_0) + \bar{\mathbf{x}}_0 + \mathbf{x}^*_0 \tag{23}$$

and by isolating $\bar{\mathbf{x}}_1$ we can obtain the linear motion equation in expression 21 without having to apply a linearization step. In a similar way, the friction constraints can also be expressed as linear constraints with respect to $\bar{\mathbf{u}}_0$

Let $m = (s_0, ..., s_N)$ denote the hybrid mode where at step j of the fixed mode MPC sub-problem the system is in the hybrid state s_j. Then, the sub-problem constraints are formulated by substituting i with s_j in equations 21 and 22. Then the optimization problem is fed into the commercial optimizer Gurobi [2] to obtain the optimal values of $(\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_N)$ and $(\bar{\mathbf{u}}_0, ..., \bar{\mathbf{u}}_{N-1})$, together with the optimal cost function. The full MPC problem solution is then approximated to be the best sub-problem solution.

As will be explained in section 6.3, we also implemented an MIQP solver to compare our FOM against. The formulation we have presented in this section has already linear constraints and quadratic cost function, so we can apply quadratic programming solvers to them. Gurobi allowed us to formulate the mode dependent constraints by using binary decision variables and a big-M notation. The resulting formulation for the motion and friction constraints is as follows:

$$\text{if } j = 0 : \begin{cases} \mathbf{x}_1 \leq (\bar{\mathbf{x}}_0 + h\mathbf{B}_{i0}(\mathbf{x}_0) \cdot \bar{\mathbf{u}}_0 + h\mathbf{f}_{i0} + \mathbf{x}_0^* - \mathbf{x}_1^*) + M(1 - z_{i0}) \\ -\mathbf{x}_1 \leq -(\bar{\mathbf{x}}_0 + h\mathbf{B}_{i0}(\mathbf{x}_0) \cdot \bar{\mathbf{u}}_0 + h\mathbf{f}_{i0} + \mathbf{x}_0^* - \mathbf{x}_1^*) + M(1 - z_{i0}) \\ \mathbf{D}_{i0}\bar{\mathbf{u}}_0 \leq \mathbf{g}_{i0} + \mathbf{1}_{2\times 1} M(1 - z_{i0}) \end{cases} \tag{24}$$

$$\text{if } j > 0 : \begin{cases} \bar{\mathbf{x}}_{j+1} \leq [(\mathbf{I} + h\mathbf{A}_{ij}) \cdot \bar{\mathbf{x}}_j + h\mathbf{B}_{ij} \cdot \bar{\mathbf{u}}_j + h\mathbf{f}_{ij} + \mathbf{x}_j^* - \mathbf{x}_{j+1}^*] + M(1 - z_{ij}) \\ -\bar{\mathbf{x}}_{j+1} \leq -[(\mathbf{I} + h\mathbf{A}_{ij}) \cdot \bar{\mathbf{x}}_j + h\mathbf{B}_{ij} \cdot \bar{\mathbf{u}}_j + h\mathbf{f}_{ij} + \mathbf{x}_j^* - \mathbf{x}_{j+1}^*] + M(1 - z_{ij}) \\ \mathbf{E}_{ij}\bar{\mathbf{x}}_j + \mathbf{D}_{ij}\bar{\mathbf{u}}_j \leq \mathbf{g}_{ij} + \mathbf{1}_{2\times 1} M(1 - z_{ij}) \end{cases} \tag{25}$$

$$z_{1j} + z_{2j} + z_{3j} = 1 \tag{26}$$

The equality constraints $a = b$ (where a and u are functions of $\mathbf{x}$ and $\mathbf{u}$) have been broken into two inequality ones $a \leq b$ and $-a \leq -b$ and each mode independent constraint $a \leq b$ has been modified into $a \leq b + M(1 - z_{ij})$ for an appropriate binary variable z_{ij}. If the sub-problem is in state i at step j, $z_{ij} = 1$ and the term $M(1 - z_{ij})$ is zero and the original inequality constrains applies. If the sub-problem is not in that state at step j, $z_{ij} = 0$ and the inequality becomes $a \leq b + M$. If the value of M is high enough (specifically, if it is higher than any possible value of the expression a - b) the inequality becomes redundant. Equation 26 enforces that the MPC sub-problems are in one and only one hybrid state at each step.

6. Simulation Interface

The simulations of Hogan's study were mainly designed for an specific choice of modes and a specific trajectory to follow. In this study, we wanted to study general properties of the FOM problem, to generalize the results to more complex pusher-slider systems and trajectories, and to compare our different FOM strategies to solve the MPC problem with other conventional methods as MIQP. In order to satisfy this needs, we decided to implement a general simulation tool. The code main structure can be described as follows:

An abstract state interface is implemented to provide the dynamics and cone constraints of the state as well as their linearized versions at a certain objective state space and control input. To define a certain experimental problem, the dynamics function and constraint equations are derived depending on a general state space variable $\mathbf{x}$ and controller action $\mathbf{u}$ and the necessary state instances are implemented and a mode is then defined by a vector of state handles. An MPC solver class it's also implemented to get an initial condition and an objective trajectory and return a control input. Each MPC solver implementation can include its own member states or modes (as well as other optimization set-up variables) and internally decides how to handle them.

At the same time, an Euler integration simulator is implemented. This simulator uses it's own state classes to describe the dynamics of the pusher slider system. These states are independent from the optimization ones so that discrepancies between model and reality can be simulated (for example the MPC

solver can use linearized dynamics or cone constraints while the Euler integrator uses the non-linearized version, which is more similar to the real experimental set-up than using linearized dynamics in both places). The Euler integration class also holds a member MPC solver that returns the control input to be applied. The integrator then computes the value of the state variable derivative $\dot{x}$ and uses a simple quadrature to approximate the next state variable value.

The controller (MPC solver) and simulator are decoupled, so they can consider different hybrid states. This allows us to consider more realistic models for the euler integration step of the simulation (we always use non-linearized motion equations for the integration step), to use different parameters in their states (to test controllers designed for a certain slider shape, μ_g, and μ_p together with a simulator set-up with different parameters, to show the robustness of the controller to uncertainties in the models).

6.1 Local, Global and MPC Costs

During the previous work at MCubelab, there was no systematic and objective way to evaluate the global trajectory resulting from the FOM control law, the exit or failure of an experiment was mainly determined in a subjective binary way. If the pusher slider didnt show appreciable divergence (at the naked eye) from the planned trajectory it was considered a success, otherwise it was a failure. In a similar fashion if the system was perturbed (by an external interaction), the success lied in recovering the original trajectory in a reasonable time determined subjectively. The main goal of the study was just to demonstrate that FOM could effectively control the pusher slider system to track previously planned trajectories, and that all this process could be done online. Because of this proof-of-concept nature, this approach was sufficient.

For this thesis we wanted a more objective way to evaluate how good a control trajectory was and this required a certain notion of metric or score for the global trajectory. We tried to keep coherence between the global and local cost functions, but removed the $\bar{\mathbf{x}}_N^T \mathbf{Q}_f \bar{\mathbf{x}}_N$ term of the cost function. As presented in section 3.3, this term is used to account for the information missing because of the finite horizon. When scoring the final trajectory, we have the full horizon information, so this term is no longer required. The obtained cost function is $S = \sum_{i=0}^{M} (\bar{\mathbf{x}}_i^T \mathbf{Q} \bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i^T \mathbf{R} \bar{\mathbf{u}}_i)$, where M is the number of euler integration steps. We will denote by local cost over time each of the $\bar{\mathbf{x}}_i^T \mathbf{Q} \bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i^T \mathbf{R} \bar{\mathbf{u}}_i$ values.

It is important to note that even if this global cost function is used score how close the obtained trajectory was from the desired one, it is not a direct indicator of the MPC solving performance. We will denote the cost function provided in section 3.3 as MPC cost. Then optimizing the MPC cost at each iteration could incur in a larger optimal cost for the global trajectory. If we are currently at the state x_i and we consider two modes M_1 and M_2 with local costs c_1 and c_2, with $c_1 < c_2$ there's no guarantee that executing the first step of the trajectory obtained from mode M_1 will translate in a smaller global cost function. Some causes for this effect are exposed and fixed in 6.4 but, even after fixing them and considering an ideal world scenario (where the models are completely accurate and there's no sensor noise), the fact that the receding finite horizon adds new information to the problem at every control iteration implies that the optimal state and control pair (x_{i+1}, u_i) computed at the control iteration i could technically lose their optimal property when the information from control iteration $i+1$ is introduced.

This effect is common in all receding finite horizon problems and there's usually no way to completely solve it. In order to evaluate the MPC solving performance of our method, some of the simulation results we expose in the following sections show a plot of the local MPC cost over time in addition to a graphical representation of the trajectory and it's global or local cost. It will be shown that, when extending a family F_1 into a family F_2 by adding more modes, the local cost of F_2 descends with respect ot F_1 in the first

bifurcation of the local MPC cost over time plot. This will be further discussed in section 6.5.

6.2 Nominal Mode

In section 5.5 we noted that, as opposed to the previous study, we do not simplify the term $f_i(\mathbf{x}^*, \mathbf{u}^*) + \dot{\mathbf{x}}^*$ in equation 16. $\mathbf{x}^*$ can be described by the same motion equations as $\mathbf{x}$ using the hybrid state friction constraints to determine the hybrid state, but it is important to realize that, once we constrain out system to be in a state i at a time instant t_j, the specific motion equation that describes this instant-state system $f_i(\mathbf{x}_j, \mathbf{u}_j)$ may not describe the instantaneous motion of the nominal trajectory. The nominal trajectory will clearly be described by $f_{i'}(\mathbf{x}_j^*, \mathbf{u}_j^*)$, but there's no guarantee that $i = i'$. The error introduced by assuming $f_i(\mathbf{x}^*, \mathbf{u}^*) + \dot{\mathbf{x}}^* = 0$ can be critical for the performance of the controller. In Figure 12 we can see the behaviour of the FOM approach when trying to track a circle using a family of modes learned using the procedure described in section 7.2. The red slider follows the nominal trajectory. The trajectory followed by its center of mass trajectory is represented with the black line. The blue slider follows the simulated trajectory and its center of mass follows the blue line. The last position of the simulated slider is highlighted and the other snapshots are faded away. The considered optimal paths the MPC considers are shown at the center of each simulated slider. The chosen path is the yellow one. As can be seen the obtained trajectory diverges from the nominal one. Furthermore, it can be seen how the optimal finite horizon trajectory computed in the MPC problem is completely different from the realized trajectory. In Figure 13 the correspondent local cost over time is shown.

The possibility of the hybrid states associated the nominal trajectory to be different from the ones associated with the real trajectory has another important implication. The perturbation state motion equation 16 has the nominal trajectory as an equilibrium point only when $i = i'$. Otherwise, if we consider $\mathbf{x} = \mathbf{x}^*$ and $\mathbf{u} = \mathbf{u}^*$, we obtain $\bar{\mathbf{x}} = \mathbf{A}_i \cdot \mathbf{0} + \mathbf{B}_i \cdot \mathbf{0} + f_i(\mathbf{x}^*, \mathbf{u}^*) - \dot{\mathbf{x}}^* = f_i(\mathbf{x}^*, \mathbf{u}^*) - f_{i'}(\mathbf{x}^*, \mathbf{u}^*)$. If none of the MPC modes is the one that the nominal trajectory follows, the solution $\bar{\mathbf{x}}_j = \bar{\mathbf{u}}_j = \mathbf{0}$ will never be feasible, and we will never be able to stabilize the planned trajectory on the nominal one, even if the initial state is on the nominal trajectory ($\mathbf{x}_0 = \mathbf{0}$). In order to force the MPC to consider the nominal mode, we can simply compute it at every MPC step (it can be easily determined applying the friction constraints to each pair $\mathbf{x}_j^*$, $\mathbf{u}_j^*$) and add it as one of the modes of the family.

6.3 MIQP and Clustered MIQP

We wanted to compare the solutions provided by FOM with traditional MPC solving techniques. In order to do that we decided to implement the MIQP technique using the commercial solver Gurobi. This solver was already used in Hogan's work in a force-based formulation prior to the final velocity-based one. This formulation was not presented in the final paper, but it allowed to formulate an MIQP problem that was able to run in an average of 3Hz. It should be mentioned that the execution time at every control iteration could range between few milliseconds to 10 seconds depending on the initial condition of the MPC. This is not a problem for simulation, but in a real control environment the frequency at which you can update the controller is not bounded by the average frequency at which you can solve the MPC, but by the worst case scenario one. Otherwise the controller may try to update the control input before it is computed by the optimizer. There may be solutions to this effect in other formulations, but for the one used in Hogan's work the MIQP controller could run at more than 0.1Hz.

We were expecting a similar behaviour in the velocity-based formulation, but it took several orders of magnitude more time to solve each iteration. In order to simulate 500 simulation steps with a finite

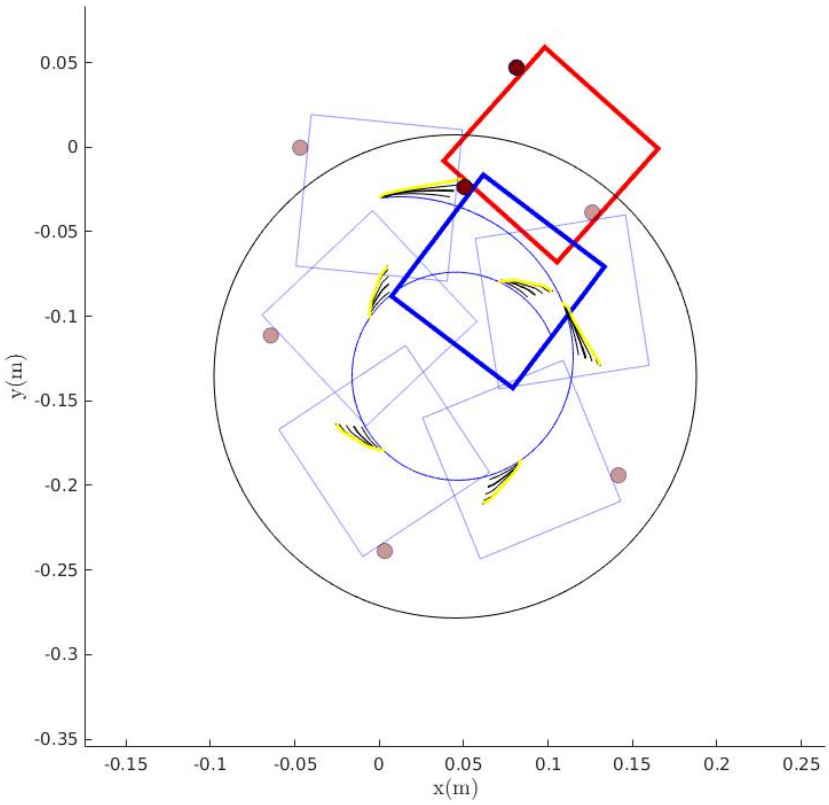

Figure 12: Snapshot from the video simulating the trajectory obtained when tracking a circular trajectory with the 8 best modes obtained using section 7.2 procedure.

horizon of 7 steps it took an average of 6 hours of execution time, which implies that the maximum control frequency we could apply would probably not surpass 0.02Hz. We were expecting to have an "ideal" MIQP result to compare FOM with but, as will be shown in section 6.5, the results obtained with a small finite horizon for the MIQP did not track properly many of the non-trivial nominal trajectories and initial perturbations we fed into it.

We looked at techniques to speed up the MIQP execution in the Gurobi website and we found out that our current big-M formulation worked better (both in execution time and in quality of the obtained trajectory) as smaller the M parameter was. As explained in 5.6 each M parameter used in a scalar inequality $Ax \leq b + M(1-z)$ must be greater than $\max_{x \in D}(Ax - b)$ (where D is the domain of possible values of the x) so that the inequality is rendered redundant when z = 0. We implemented a version of MIQP that given a vectorial constraint as the ones used in Hogan's formulation $A\vec{x} \leq \vec{b} + M(1-z) \cdot (1, ..., 1)^T$ replaces the arbitrary constant M by a vector $\vec{M}$ where each component's value is $M_i = \max_{x_i \in D}(A_i x_i - b_i)$. We were not able to improve the execution time, but we were able to obtain better results for the MIQP. The results showed in the next section for the MIQP will use this optimized big-M formulation.

In order to compare FOM to other methods using a longer time horizon, we implemented another MPC

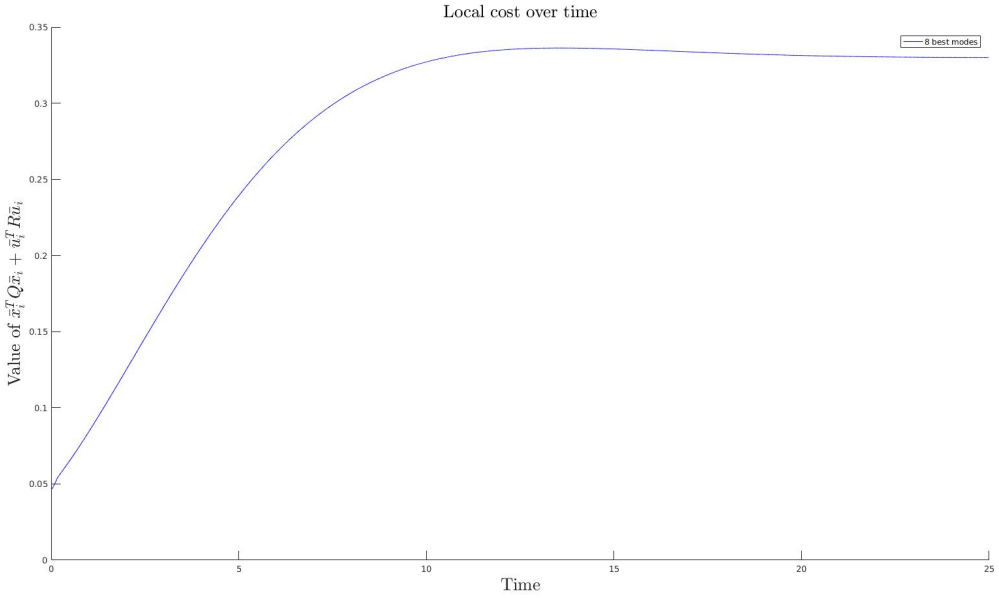

Figure 13: Local cost over time associated to the trajectory in Figure 12

solving method based on MIQP that we called Clustered MIQP. Given a standard MPC formulation, this method works in the following way:

- Consider a clustering factor c, a finite horizon $N \cdot c$ and a set of h hybrid states $\{H_1, ..., H_h\}$.
- The finite horizon is then divided into N clusters of c steps: $(C_1, ..., C_N)$.
- The state space is extended in a similar way as in section 5.6 with 3 binary variables per cluster for a total of $3N$ variables.
- Each variable z_{1i} z_{2i}, and z_{3i} is then associated with the hybrid state of the cluster C_i (sticking, sliding down and sliding up respectively).
- The constraint $z_{1i} + z_{2i} + z_{3i} = 1$ is enforced for all i.
- The problem constraints are formulated with big M notation as in section 5.6 and the problem is fed into a Gurobi MIQP solver.

As we exposed in section 3.3, the MIQP formulation doesn't escalate well with the number of binary variables added and clustered MIQP allows us to reduce the binary variable dimensionality of the optimization problem by keeping the same state during all the steps of the same cluster. The execution time of this new problem is similar to the execution time of a MIQP problem of N steps but the controller inputs can be different at each of the Nc steps, providing a more flexible trajectory than the one given by the MIQP problem of N steps.

We intended to use this method to approximate MIQP problems with long horizons by using few clusters and we expected to obtain a better trajectory than the ones obtained with FOM. As will be discussed later,

the results of this MIQP approximation were usually worse than the ones obtained in the FOM with the original modes from Hogan's study. In section 6.5 we provide a possible explanation for this behaviour.

6.4 Chameleon Mode

In the previous FOM study there was no guarantee of improvement in the obtained trajectory by considering new modes in a previously used family. It needs to be clarified that, even if in the previous study there was no global score function implemented, this effect could be so big that it could be observed with the naked eye. An illustrative example would be the following. A family containing only the sticking mode $F_1 = \mathbf{S}$ would greatly outperform the extended family $F_2 = \mathbf{S} \cup (L, R, S, \overset{N-2}{\cdots}, S)$. On the other side a further extension $F_3 = F_2 \cup \mathbf{R}_1$ would outperform the two previous families. This behaviour can be shown in Figure 14.

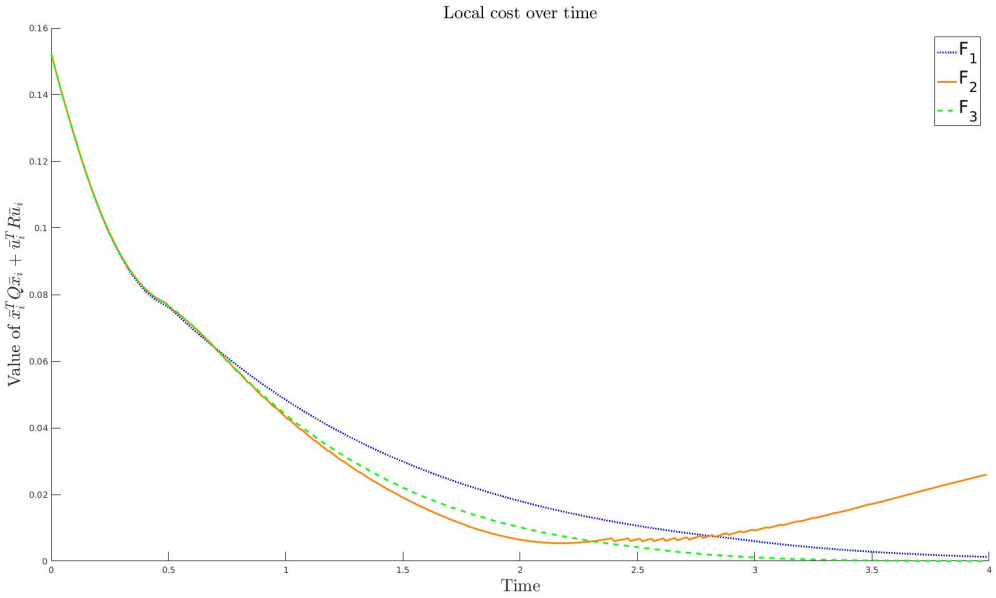

Figure 14: Local cost over time for families F_1, F_2, and F_3 obtained following a straight line on the x axis and using an initial condition $\mathbf{x}_0 = (0, 0.05, -\frac{\pi}{6}, 0)^T$.

The performance decrease experienced in F_2 can be explained by the fact that the FOM optimizer can choose optimal trajectories that will never be able to be applied. For example, the optimal control solution at a certain optimization iteration can be to choose the second mode of F_2: start with an L state for the first time step to then choose a R step and finally choose an S step for the rest of the finite time horizon prediction. The capability of sliding left and right should allow the pusher to take more aggressive turns by sliding to one of the slider edges and then slide back to the center to properly track the straight line. The main problem is that the control law for the controller is to execute the only th first step of the optimal trajectory, so the system will be in an L state for one time step and then the control law will be recomputed. As no mode starts with a sliding right state the slider can never go back to the central position.

The conclusion these experiences led us to was that, using Bersekas' terminology, some families did not

have the sequential improvement property. What's more, the results seemed to indicate that composition or expansion of families that apparently had sequential improvement could lead to losing the property.

Solving the MPC using the FOM approach has similarities with the roll-out algorithm presented in their study: Each of the modes can be considered a heuristic and the strategy of recomputing the control policy at a certain frequency by using the same heuristics can be viewed as a roll-out-like technique. In fact, we could easily reformulate any family of our FOM approach to behave as their roll-out by increasing the finite horizon by one step and substitute each of the original modes by a triplet of copies preceded with an S, L, and R state respectively. We decided not to use an exact roll-out because, as we will show in this section, we were able to find a strategy to guarantee a sense of improvement without having to increase the number of MPC sub-problems to solve by a multiples of the number of possible states.

It's important to note that the concepts of sequential consistency or optimality cannot be applied to our case without some previous modifications, because of the existence of a receding finite time horizon, but for every FOM problem we can formulate an associated Full Horizon Family of Modes (FFOM) problem as follows: (TODO: Define it).

Definition 6.1. FFOM:
Given a FOM problem defined by a finite horizon of N steps, a set H of possible hybrid states, a cost function J, and a family $F^{(0)} = \{M_1^{(0)}, ..., M_k^{(0)}\}$ containing k modes, where $M_i^{(0)}$ is a general mode described as $M_i^{(0)} = (s_{i,1}, s_{i,2}, ..., s_{i,N})$, with $s_{ij} \in H$. Furthermore, consider a cost function $J_k = \sum_{i=k}^{N} \left(\bar{\mathbf{x}}_i^T \mathbf{Q} \bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i^T \mathbf{R} \bar{\mathbf{u}}_i\right)$ (defined using the $\mathbf{Q}$ and $\mathbf{R}$ matrices from the FOM cost function J) and the prefixes of each of the modes of the family $M_i^{(j)} = (s_{i,1}, ..., s_{i,N-j})$. Then the associated FFOM problem is defined iteratively as follows:

1. Solve the FOM-based MPC problem defined by $F = F^{(0)}$ and $J = J_0$, using x_0 as the initial condition.
2. Use the controller $\bar{\mathbf{u}}_0$ from the optimal solution of the problem solved in the previous step to compute the value of $\bar{\mathbf{x}}_1$. Under an ideal world assumption (that is, assuming that our motion equations model is accurate) the value of $\bar{\mathbf{x}}_1$ will correspond to the same value obtained in the optimal trajectory of the MPC.
3. Replace F by $F^{(1)} = \{M_1^{(1)}, ..., M_k^{(1)}\}$, J by J_1, and the initial condition by $\bar{\mathbf{x}}_1$ and go to step 1, solving this time an MPC problem of $N - 1$ steps.
4. Repeat until the horizon has only one step.

This formulation represents how the FOM problem would behave be if we had enough computation power to cover a finite horizon that reaches the desired final time step. Using it, we can study the FFOM problem associated to the example families presented at the beginning of this section. For simplicity, we will consider the realized states and control inputs to describe the considered trajectories.

Example 6.2. $\mathbf{F}_1$ is sequentially consistent in it's FFOM version and assuming an ideal world:

Consider that at a certain control iteration there's a full horizon of N steps to reach the end of the trajectory. The only considered mode, $\mathbf{S}$, get the optimal path to reach the goal while keeping the sticking contact at all times, obtaining a sequence of position states $\mathbf{x}^{\mathbf{S}}(\mathbf{x}_0) = (\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, ..., \mathbf{x}_{1,N})$ and control inputs $\mathbf{u}^{\mathbf{S}}(\mathbf{x}_0) = (\mathbf{u}_{0,1}, \mathbf{u}_{1,1}, ...\mathbf{u}_{1,N-1})$ for the optimal path from the initial position to the goal. Then, the controller will execute $\mathbf{u}_0$ for one control step and the pusher slider system will move to $\mathbf{x}_1$. The only heuristic it considers in the next optimization step is to stick until reaching the goal, so the optimal trajectory obtained will be $\mathbf{x}^{\mathbf{S}}(\mathbf{x}_1) = (\mathbf{x}_{1,2}, ..., \mathbf{x}_{1,N})$ and $\mathbf{u}^{\mathbf{S}}(\mathbf{x}_1) = (\mathbf{u}_{1,1}, ...\mathbf{u}_{1,N-1})$, being sequentially consistent.

Example 6.3. On the same conditions, $\mathbf{F}_2$ is not sequentially consistent nor improving:

Let's call the second mode of family $\mathbf{LR}_1$. If this mode is chosen at a certain time step, giving an optimal trajectory $\mathbf{x}^{\mathbf{LR}_1}(\mathbf{x}_0)$ and control $\mathbf{u}^{\mathbf{LR}_1}(\mathbf{x}_0)$. In the next step, a mode $\mathbf{R}_1$ is required in the family to consider the continuation of the previous optimal trajectory, which consists in sliding right and then sticking until the end of the trajectory. As it is not the case, the new FFOM step will not consider the previously optimal trajectory and, if no relation can be proven between the trajectories considered in the current step and the ones considered in the previous step, neither consistency nor improvement cannot be guaranteed.

Consider that, at a given step of a FFOM problem, the initial position is given by $\mathbf{x}_0$ and the optimal trajectory is determined by the mode $M_1^{(0)} = (s_{1,1}, s_{1,2}, ..., s_{1,N})$. If the family contains a mode whose maximal prefix corresponds to the maximal suffix of $M_1^{(0)}$, $M_2^{(0)} = (s_{1,2}, ..., s_{1,N}, s_{2,N+1})$ then sequential improvement will be guaranteed at the next iteration step. That is, if the trajectory provided by M_1 at the first step is $\mathbf{x}^{M_1^{(0)}}(\mathbf{x}_0) = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ and $\mathbf{u}^{M_1^{(0)}}(\mathbf{x}_0) = (\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N-1})$ and the one provided by M_2 at the second step is $\mathbf{x}^{M_2^{(1)}}(\mathbf{x}_1) = (\mathbf{x'}_2, ..., \mathbf{x'}_N)$ and $\mathbf{u}^{M_2^{(1)}}(\mathbf{x}_1) = (\mathbf{u'}_1, ..., \mathbf{u'}_{N-1})$, then

Lemma 6.4. *$J_1(\mathbf{x}^{M_2^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_2^{(1)}}(\mathbf{x}_1)) = J_1(\mathbf{x}^{M_1^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_1^{(1)}}(\mathbf{x}_1))$:*

As $M_2^{(1)}$ is the maximal suffix of $M_1^{(0)}$ the continuation of the trajectory provided by M_1 on the first step is feasible in the mode M_2 during the second step so, as the trajectory given by M_2 is optimal, $J_1(\mathbf{x}^{M_2^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_2^{(1)}}(\mathbf{x}_1)) \leq J_1(\mathbf{x}^{M_1^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_1^{(1)}}(\mathbf{x}_1))$

If we assume $J_1(\mathbf{x}^{M_2^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_2^{(1)}}(\mathbf{x}_1)) < J_1(\mathbf{x}^{M_1^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_1^{(1)}}(\mathbf{x}_1))$ we can easily reach a contradiction. Let's define $\mathbf{x}^* = (\mathbf{x}_1, \mathbf{x'}_2, ..., \mathbf{x'}_N)$ and $\mathbf{u}^* = (\mathbf{u}_0, \mathbf{u'}_1, ..., \mathbf{u'}_{N-1})$. This is a feasible trajectory in $M_1^{(0)}$ but $J_0(\mathbf{x}^*, \mathbf{u}^*)$ can be expressed as:

$$J_0(\mathbf{x}^*, \mathbf{u}^*) = \mathbf{x}_0^T \mathbf{Q} \mathbf{x}_0 + \mathbf{u}_0^T \mathbf{R} \mathbf{u}_0 + J_1(\mathbf{x}^{M_2^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_2^{(1)}}(\mathbf{x}_1))$$

and we can conclude that $J_0(\mathbf{x}^*, \mathbf{u}^*) < \mathbf{x}_0^T \mathbf{Q} \mathbf{x}_0 + \mathbf{u}_0^T \mathbf{R} \mathbf{u}_0 + J_1(\mathbf{x}^{M_1^{(1)}}(\mathbf{x}_1), \mathbf{u}^{M_1^{(1)}}(\mathbf{x}_1)) = J_0(\mathbf{x}^{M_1^{(0)}}(\mathbf{x}_0), \mathbf{u}^{M_1^{(0)}}(\mathbf{x}_0))$, reaching a contradiction.

Let's now consider an extension state E. That is, a default state to apply mode extensions. We can then define a mode suffix operation as follows.

Definition 6.5. Mode Suffix Operation:

Given a default extension state E, applying the suffix operation to mode $M_1 = (s_1, s_2, ..., s_N)$ results in $M_2 = suff(M_1) = (s_2, ..., s_N, E)$

This definition allows us to announce the following corollary from lemma 6.4.

Corollary 6.6. *A family F closed by a suffix operation is sequentially improving in the FFOM problem:*

At any step i of the approximated MPC, a certain mode M will provide the optimal trajectory. As $suff(M) \in F$ we can use lemma 6.4 to guarantee that the trajectory provided by $suff(M)$ in step $i+1$ is as good as the one currently provided by M. As the approximated MPC considers the optimal trajectory between all the considered modes, the computed trajectory at $i+1$ will be at least as good as the one computed in i.

Example 6.7. On the same conditions, $\mathbf{F_3}$ is sequentially improving:

F_3 is closed by the suffix operation defined using S as the extension state, so we can use corollary 6.6 to guarantee sequential improvement in the FFOM.

We considered important to guarantee sequential improvement for the FFOM problem even taking into account that our problem is not in the ideal world (modelling errors and different frequencies for the optimization problem and the controller can lead the state obtained when applying u_0 to $\bar{x_0}$ to be different from x_1) and that we don't have a full horizon. As we showed in the examples above, the simulation results were in agreement with these theoretic results as the families that implicitly guaranteed sequential improvement in their FFOM version tended to outperform the ones that did not.

In order to guarantee it we decided first decided to only use families that were closed by the suffix operation obtained using **S** as the extension mode. The problem with this approach is that for any mode $M = (s_1, ..., s_N)$ you add to a family F you need to include all of its possible **S**-extended suffixes. Different modes required adding a different number of suffix modes. For example, an **S** mode required no extra modes (as it is itself closed by the suffix operation) and for a $\mathbf{R}_n$ one you only need to extend the family with an **S** mode and all the $\mathbf{R}_i$ modes for $i < n$. Even taking this into account, we considered adopting this strategy constrained the problem of family design more than we wanted to and added more extra nodes that was strictly necessary, so we decided to adopt another strategy. This translated into changing from static families to dynamical ones.

(TODO: Explain why **S**)

The strategy we used consisted in including a dynamic mode that we called chameleon mode. If, at a certain step i, the optimal FOM mode is mode M, the chameleon mode is defined in step $i+1$ as $suff(M)$ (using, as before, **S** as the extension state). This guaranteed sequential improvement by applying lemma 6.4 in an analogue way as in corollary 6.6.

The chameleon mode functionality was added in the simulation interface and it's performance was tested in simulation. As we will show in section 6.5, implementing this functionality resulted in an improvement of the sequential behaviour of the FOM algorithm. Being able to safely add arbitrary new modes to the FOM family without fear of losing its sequential properties allowed us to develop an strategy to learn a good family set-up for a general trajectory tracking problem, as will be shown in chapter 7.

Given the good properties of the chameleon mode, we decided to include it in the FOM formulation so, from now on,we will use FOM to refer to the family of modes with chameleon mode implementation, unless specified otherwise.

6.5 Simulation Results

In order to compare FOM and MIQP we considered necessary to use the same finite horizon length in both formulations to have a fair comparison. As explained in section 6.3, the computational cost was much higher than we expected. The maximum finite horizon length we could afford to run simulations in a reasonable time was 5. This value of the finite horizon was too short to control most of the non-trivial trajectories properly. Figure 15 shows the trajectory obtained from MIQP while trying to follow a circular trajectory and it can be seen that it clearly diverges.

We tested our chameleon mode implementation by comparing the performance of families of modes that are not sequentially consistent in their full horizon formulation with and without the chameleon mode with satisfactory results. In Figure 16 we show the example presented in section 6.4. All the simulation parameters are kept the same except for the usage or non usage of the chameleon mode. As

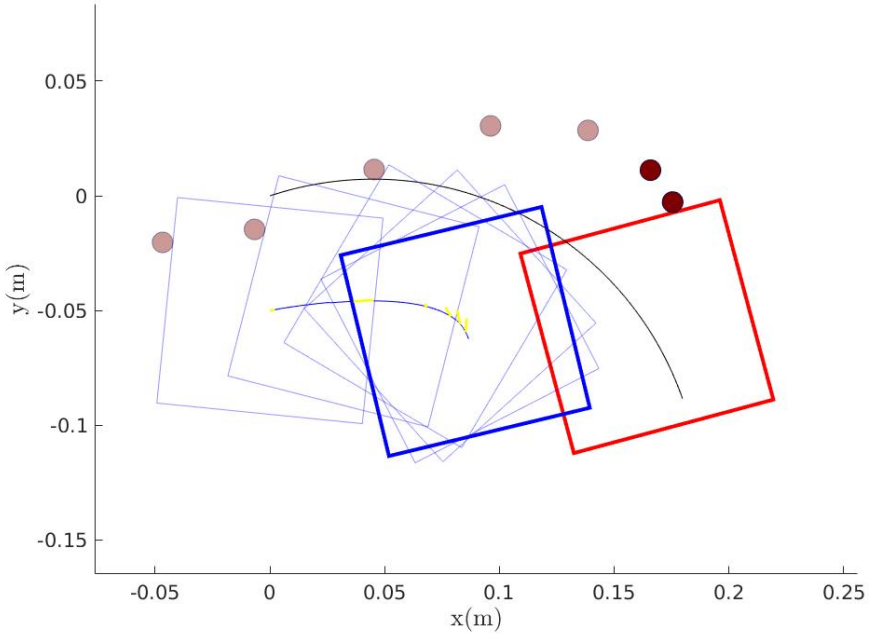

Figure 15: Trajectory obtained when tracking circular trajectory defined by pushing perpendicularly to the slider with $p_y = 0.0249$. Initial position $\mathbf{x}_0 = (0.0, -0.05, -0.1, p_y)^T$.

can be observed, the results correspond with our expectations: for the family F_2 the chameleon mode implementation successfully tracks the trajectory while the implementation without it failed to do it, as we showed previously.

TODO: Talk about the zoomed in figure, to talk about the glory of roll-out.

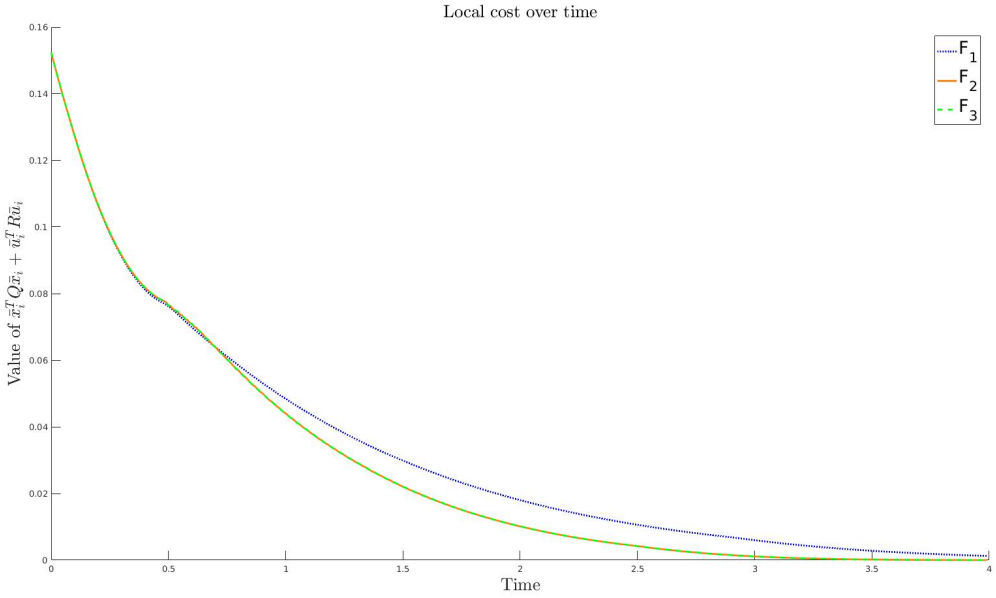

Figure 16: Local cost over time for families F_1, F_2, and F_3 obtained following a straight line on the x axis and using an initial condition $\mathbf{x}_0 = (0, 0.05, -\frac{\pi}{6}, 0)^T$.

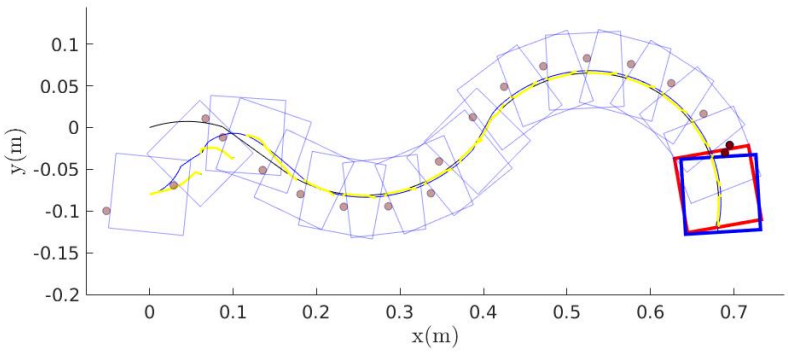

Figure 17: General trajectory followed by using the original modes from Hogan's study. Finite horizon of 35 steps.

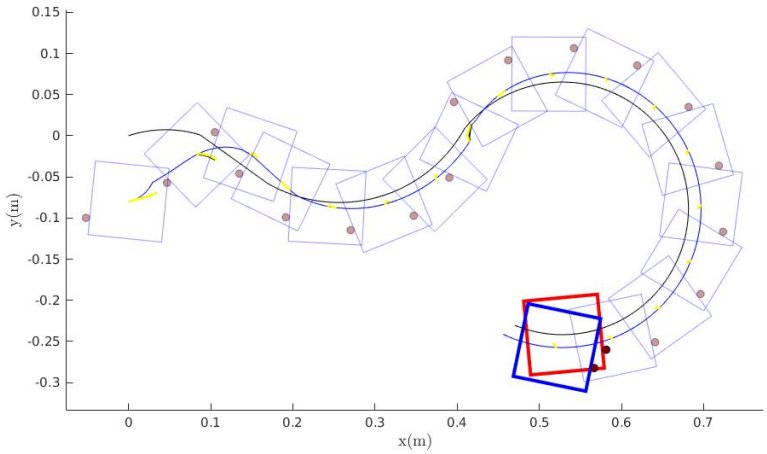

Figure 18: General trajectory followed by using the original modes from Hogan's study. Finite horizon of 15 steps.

Finally, we compared clustered MIQP with five clusters of 7 steps with a FOM approach using three modes family formed by the S, U_1, and D_1 modes.

7. Mode Selection Protocol

In the previous sections it is explained how we addressed the sequential properties of the FOM formulation and provided a metric and other MPC solvers to compare FOM to. At this point of our study, we were able to adress our next objective: to optimize the selection of the family modes for the FOM formulation.

We considered multiple optimization criteria and relations to study and we had to adopt a set of assumptions to be able to obtain some initial results. For our first study, we tried to learn the modes to follow a single fixed straight line trajectory. Each optimization problem would be parametrized by a fixed finite time horizon (characterized by a number of steps N) and a fixed number of modes K a family could contain. The initial condition $\bar{x_0}$ was assumed to be a random variable following a certain probability distribution. The cost of the control trajectory C was thus also a random variable of unknown distribution parametrized by the optimization problem parameters. Specifically given a set of K modes $\{m_1, ..., m_k\}$, $C \sim D(m_1, ..., m_k)$ for a given distribution law D. Ideally, we would like to choose the the set of modes that minimizes $\bar{C}$.

We made three main assumptions:

Assumption 7.1. Local Cost Sufficiency: *Optimizing the modes to reduce the local costs obtained while solving each MPC iteration would be sufficient to obtain a low global cost.*

Assumption 7.2. Locally Invariant Trajectories: *Given a local inertial frame F centered on the slider center of mass moving and rotating with it and given an infinite objective trajectory $C(t)$ for $t \in (-\infty, \infty)$, then the $C(t)$ would be time invariant in frame F (for example an infinite circle loop or an infinite straight line).*

Assumption 7.3. Single Step Sufficiency: *A single control iteration of the trajectory tracking problem was sufficient to learn which modes where optimal, if the appropriate distribution for $\bar{x_0}$ was chosen.*

The motivation and justification for these assumptions will be explained in 7.1, as well as some other implicit assumptions contained in it. In 9 we will discuss some of the studies we would have conducted and how we would have modified these assumptions if we disposed of more time.

(TODO: Make sure I discuss them)

7.1 Structure Study And Systematic Approach

Given a finite horizon of N steps and H states and considering M the set of possible modes obtained by concatenating N of these hybrid states, then $|M| = H^N$. Let the set of possible families containing K of these modes be F. Then $|F| = \binom{|M|}{K} = \frac{H^N!}{K!(H^N-K)!}$. Even with a low number of states this number grows quickly as N and K increase.

If we tried to directly tackle the optimization problem without any assumption or knowledge of its structure, we would need to solve several trajectory tracking problems for each of these possible families, because the families could be correlated between themselves. A simple example that will probably clarify this effect is that a D_1 or U_1 family will probably perform poorly alone because if the pusher slides away from the center of the slider face it will never be able to go back to it and control near the edges of the face is less versatile than on the middle (for example, while steeper turns can be taken in the edges of the face, changing rotation direction is much more difficult, at some point not even possible without sliding to the other side).

Solving many different initial conditions for all possible mode combinations with a long enough finite horizon would take too much time even to learn the optimal modes to track a single trajectory. We didn't want to use any simplification that would be specific to our pusher-slider system, because we wanted to provide systematic tools to use the FOM approach in general problems the control community may face. This led us to use a set of general assumptions to simplify the problem and devise a systematic protocol to study the structure of each problem and use different optimization/learning techniques depending on the structure.

We mentioned the local cost sufficiency assumption again here, but this assumption was already taken when deciding to use a finite horizon MPC formulation to control the pusher-slider system. The mode selection will determine the performance of our local FOM solver and the relation between local MPC performance and overall performance and the validity of this assumption were already discussed in 6.1 and 6.5.

The locally invariant trajectories assumption motivation will be explained later in 7.3 (TODO: Make sure I don't forget to explain it)

The main motivation for the single step sufficiency assumption has to be understood under the previous assumption. The locally invariant property implies that the only difference between an MPC step and another is their initial condition random variable $\bar{x}_i$. Consider a distribution law X that focusses more on covering a wide range of possible values of any $\bar{x}_i$ rather than on their frequency as, for example an appropriate uniform distribution. We can then optimize the modes to reduce the expected value of $C(\bar{x}_0)$ of a single controlling step tracking problem, where $\bar{x}_0 \sim X$. Optimizing the mode selection this will simultaneously reduce the local costs expected value for all $\bar{x}_i$ implicitly assuming that each of these values are equally distributed.

This implicit assumption is clearly false. The fact that some trajectories converge is already a counterexample of it. We consider assuming a uniform to describe all $\bar{x}_i$ is a conservative approach, as will consider rare big disturbances (that can usually only happen with an external perturbation or a bad initial position) to be equally likely to appear at any time step as small disturbances characteristic of proper control around the objective trajectory. We expected this approach would be sufficient to provide a mode selections that would be robust enough to track a wide range of trajectories and initial conditions, but that could fail to capture some feedback relations between modes. This will be further discussed in 9.

These assumptions result in a great cost reduction and simplification of the learning algorithm because the optimization problem can be now be easily written as a function of $\bar{x}_0$:

Corollary 7.4. The optimization problem to select the FOM modes can be expressed as:

$$\min_{m_1,\dots,m_K \in M} \mathbb{E}(C(\bar{x}_0; m_1, \dots, m_K)) = \min_{m_1,\dots,m_K \in M} \mathbb{E}(\min_{i=0}^{K} C_i(\bar{x}_0))$$

Where C_i is the cost associated to mode m_i in the MPC optimization. This further implies that solving the MPC problem for all the possible sets of K modes is no longer necessary. If $C_i(\bar{x}_0)$ is computed for every mode $m_i \in M$, then $C(\bar{x}_0; m_1, \dots, m_K)$ is defined for any set of K modes by corollary 7.4.

This algorithmic cost reduction was not enough to face the full learning problem. We no longer had to study the full space of families F, but we still had no alternative to explore the full space of modes M, even if we just try to obtain a local minimum. Many local optimization techniques can avoid considering the full space it by relying on the structure of the optimization problem. For example, when the parameters are on a continuous metric space, the concept of gradient arises and methods as gradient descent can be used. Some structures can also be used to guarantee stronger properties as in convex problems, were local solutions are also global.

Considering all these aspects, we devised a systematic approach to study the structure of a general MPC problem and optimization techniques. We then provided a set of optimization techniques to be applied to obtain an approximate solution for the optimization problem. It has to be noted that the studies and techniques we will present are not proofs of a certain structure and thus, the techniques do not guarantee any kind of optimal solution. We considered proving any of these properties in an specific problem to be an object of research study in itself and it escaped the goal of this thesis, as we wanted to provide general simple tools to expand the FOM usage in the manipulation community.

TODO: Maybe get rid of all this last part, never been a fan of it.

- Finite horizon increase sensibility:
- Mode metric study: In section (TODO:reference it) we provide and discuss a mode metric. We wanted to study if there was a continuity-like property (if we bound the bound distance between modes we can bound to some extend the distance of their costs). Having this property would allow to
- Cost distribution study: If all have the same distribution, fewer samples would be required, or at least we could have boundaries on the error.

7.2 Simulation Results. Simple trajectory learning

For our initial study, we tried to learn the best modes for single locally invariant trajectories. As proposed in the previous section, we started by obtaining 1000 samples of $\bar{x}$ from a uniform distribution in a cube in $\mathbb{R}^4$ the $\bar{x}$ and $\bar{y}$ disturbances on the x and y position of the slider center of mass were bounded between 0.5 and 0.5 meters. The slider angle disturbance $\bar{\theta}$ was bound between -$\frac{\pi}{4}$ and $\frac{\pi}{4}$ and the value of p_y could lay in any point on the slider surface. For each initial condition the tracking problem was solved for all the possible modes with a small finite horizon of 5 steps. This implied 243 possible modes to be solved for each of the 1000 initial conditions, for a total of 243.000 single step tracking problems. The cost functions $C_i(\bar{x}_j)$ were computed for each mode-initial condition pair. In Figure 19

We were concerned that the initial condition had a lot of effect in the overall cost of the mode-condition pair, because higher or smaller disturbances would lead to higher or smaller overall cost respectively, independently of the mode chosen, and the simulation results confirmed it, as can be seen in Figure 19. We wanted to reduce the impact of the initial condition in the range of possible values of the local costs to obtain comparable experiments, so we decided to standardize the results obtained within each experiment. That is, given $C_i(\bar{x}_j)$ we computed $\bar{C}_i = \frac{\sum_{j=0}^{243} C_i(\bar{x}_j)}{243}$ and $\sigma_i = \sqrt{\sum_{j=0}^{243} \frac{(C_i(\bar{x}_j) - \bar{C}_i)^2}{242}}$ and we used them to compute $\hat{C}_i(\bar{x}_j) = \frac{C_i(\bar{x}_j - \bar{C}_i)}{\sigma_i}$. The resulting standardized costs can be observed in Figure 20.

We were not able to determine a clear probabilistic distribution for $\hat{C}_i(\bar{x}_j)$ so we decided to minimize the statistical mean for each distribution $\bar{C}(\bar{x}; m_1, ..., m_K) = \frac{\sum_{j=0}^{N} \min_{i=0}^{K} C_i(\bar{x}_j)}{N}$.

Choosing the set of K modes that minimizes the desired expression can depend exponentially on K, so we used a greedy upper bound approximation to compute it. The greedy approach was defined recursively as follows:

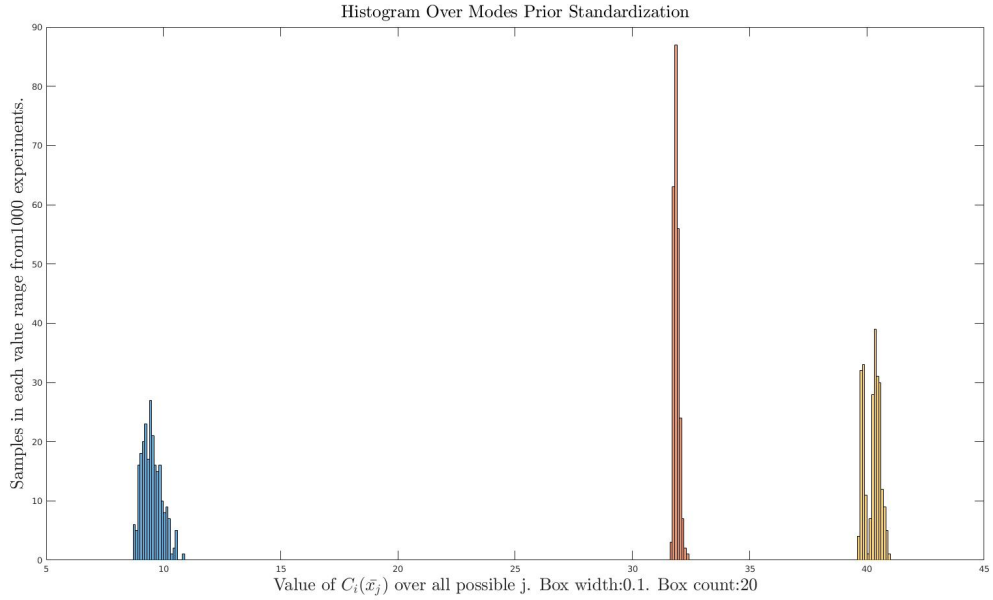

Figure 19: Example histogram of $C_i(\bar{x}_j)$ of the local costs obtained while tracking a horizontal line for 1000 initial conditions. The costs are clearly non-comparable

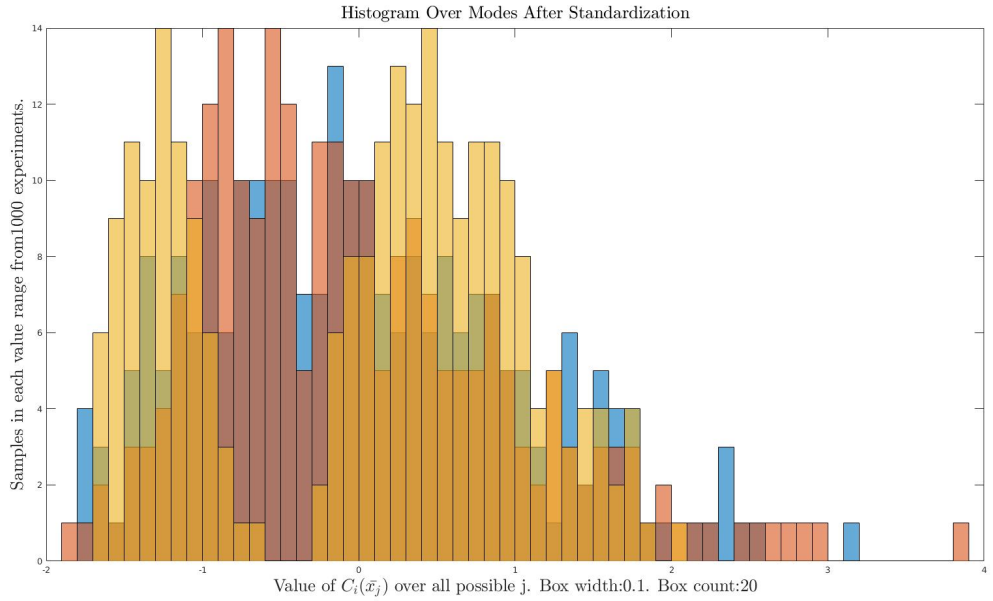

Figure 20: Example histogram of $\hat{C}_i(\bar{x}_j)$ for the same simulation set-up as in Figure 19. The costs are now comparable

Lemma 7.5. Upper bound to $\bar{C}(\bar{x}; m_1, ..., m_K)$: *(TODO: Define)*

A dynamic programming algorithm was then implemented to simulate the recursion. Furthermore, we determined the lower bound of $\bar{C}(\bar{x}; m_1, ..., m_K)$ by computing the minimum over all the modes for each experiment and computing the statistical mean. $\min_{m_1,...,m_K \in M} \mathbb{E}(\min_{i=0}^{K} \hat{C}_i(\bar{x}))$. By comparing the evolution of our computed upper bound with the lower bound we can see that the approximate algorithm quickly converges to the optimal solution (see Figure 21) this properly captures relations between modes. The algorithm can choose modes that, even if they would not provide small means by themselves, can greatly increase their performance by associating with modes that specialize in initial conditions where they fail to give a good result. (TODO: Explain Better. For example say that the min mean for one is -1.6 and that it get's closer)

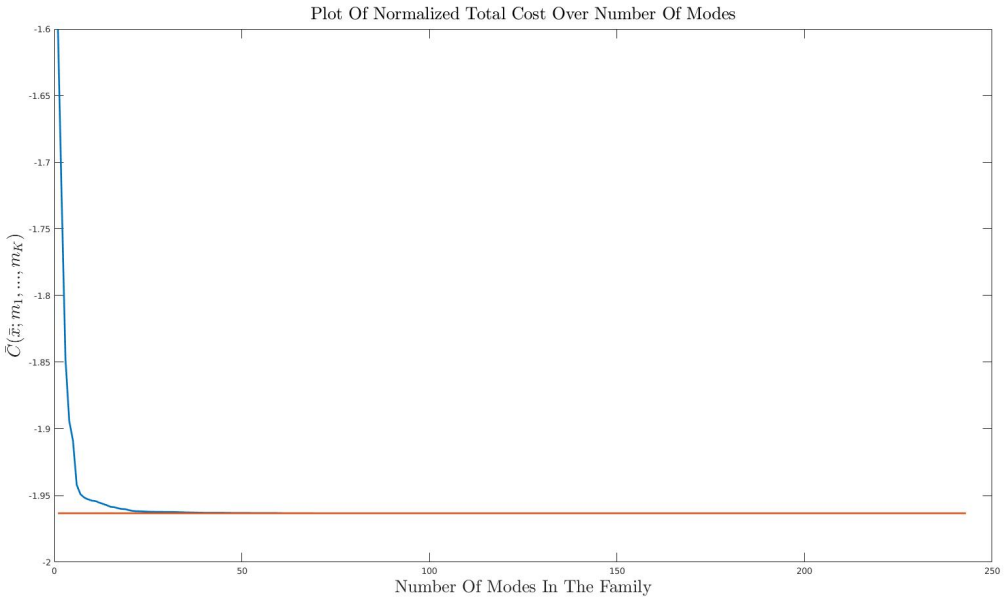

Figure 21: Plot of our greedy approximation to the minimum value of $\bar{C}(\bar{\mathbf{x}}; m_1, ...m_K)$ obtained by using families with K modes. The horizontal bar shows the optimal lower bound obtained when all the families are used. It's exact value is -1.9633.

We then proceeded to follow the strategies cited in the previous section. We successfully computed an approximation of the best modes for a 35 step length horizon by iteratively selecting the 5 best modes for $5 \cdot i$ steps for $i \in [1, 6]$ and extending them to $5 \cdot (i + 1)$ steps by adding all the possible tails to the previous best modes. Experimental results showed that state differences were more important when they occurred near the beginning of the mode than the end. This motivated us to compare the results obtained by following the full iterative procedure and the results obtained by computing the best modes for a short horizon and extending it with a long simple tail.

(TODO: add pictures of the comparisons)

The results obtained showed that adding a simple sticking tail to the best K modes chosen for a short trajectory was enough to successfully track a trajectory. (TODO: study with different tail)

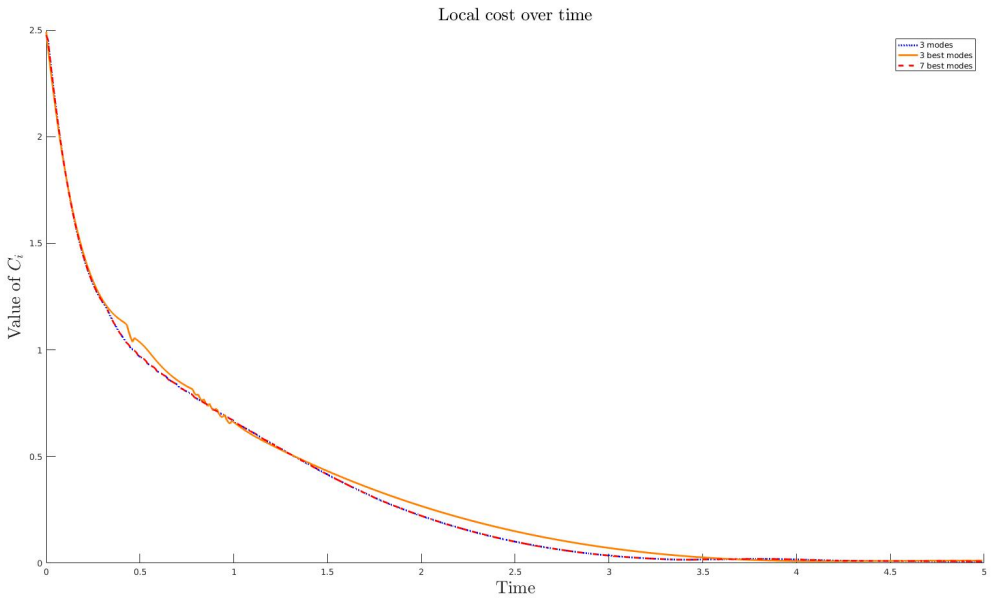

Figure 22: Comparison between the original FOM from Hogan's study and the families obtained by the learning algorithm. The initial condition is $\bar{\mathbf{x}}_0 = (0, .05, \frac{\pi}{6}, 0)^T$. The vertical axis correspond to the optimal value of the finite horizon MPC over time.

(TODO: add graph with this)

Experimental results also showed that the improvement obtained by adding new modes decreased exponentially (TODO: check the proper name for this) before reaching the optimal value, as can be seen in Figure **??**. After a few modes, the results are highly stable and the changes in the cost function are small.

Finally, we tested the robustness of the method by tracking circular trajectories with the best modes obtained for a straight line. The behaviour was completely different if the initial condition was outside or inside of the tracking circle. As can be seen in (TODO) few modes were sufficient to properly track trajectories with high external disturbances. On the other hand, small internal disturbances could lead to solutions with a steady state error that ended tracking a circle internal to the one determined by the objective trajectory if few modes were used. Progressively adding modes steadily reduced the effect.

TODO: Circle with circle modes

7.3 Dynamic FOM. Theory and results

The assumption of local invariance for the objective trajectories limited the range of application of the FOM approach as presented until now. In the previous section we could show a certain degree of robustness when using the modes learned on a simple straight line to track other trajectories, but the number of modes per family had to be increased while tracking circular trajectories in order to achieve the same results than while tracking straight lines. TODO: Check that it is true

In section 7.1 we mentioned that we had a motivation to use locally invariant trajectories before obtaining the simulation results from the previous section. This motivation was to train the FOM method

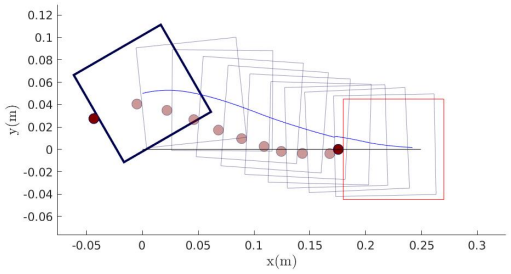

Figure 23: Trajectory obtained with the family from Hogan's work. Overall cost: 11.8847

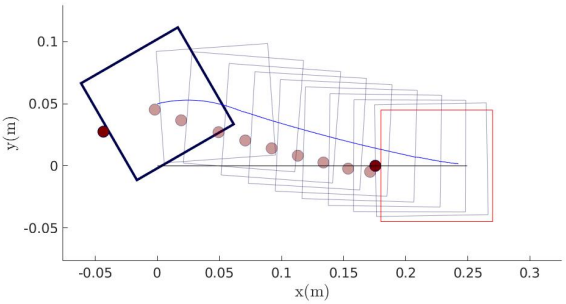

Figure 24: Trajectory obtained with the best combination of three modes computed using our learning procedure. Overall cost: 11.8664

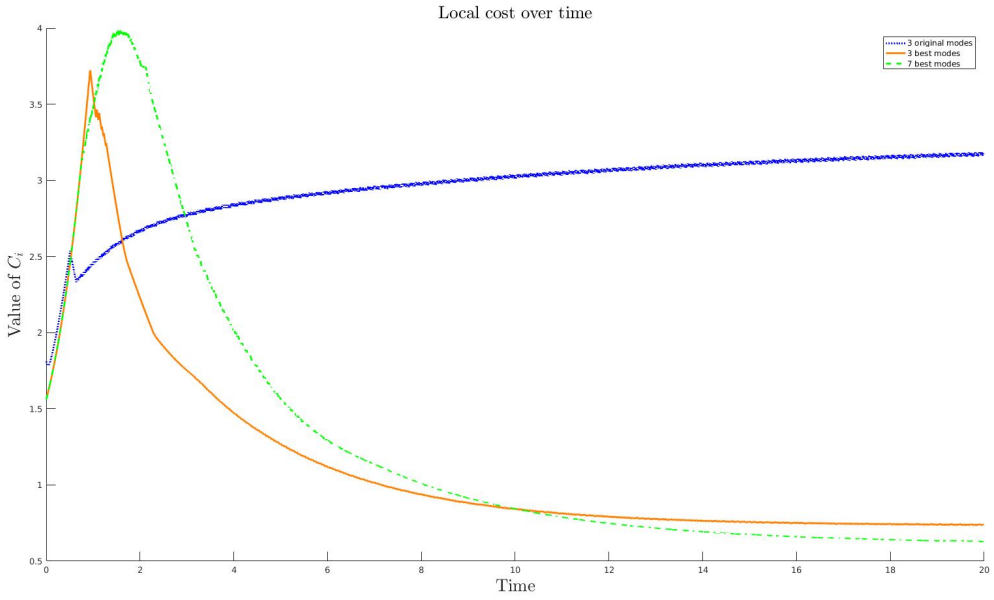

Figure 25: Analogue to Figure 22 for a circle trajectory with a internal disturbance. The initial condition is $\bar{\mathbf{x}}_0 = (0.0, -0.03, -0.1, 0.0249)^T$

on several base locally invariant trajectories that could describe most of the situations the tracking problem would face (For the simple contact pusher-slider system this consisted in a straight line and several circular trajectories of different curvatures). Then we wanted to learn the best k modes for every base trajectory. To track complex trajectories, we would approximate them as a piece-wise composition of base ones and at each controller step we would use the best modes for each base trajectory depending on the base trajectory

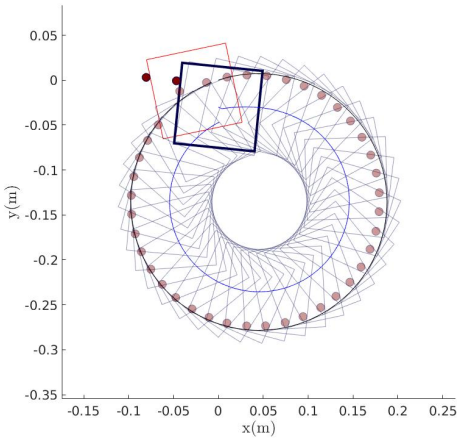

Figure 26: Trajectory obtained with the family from Hogan's work. Overall cost: 195.2719

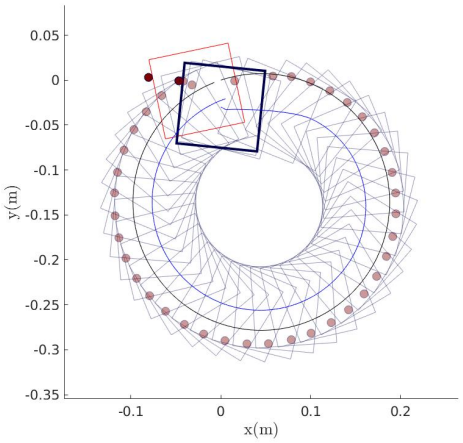

Figure 27: Trajectory obtained with the best combination of three modes computed using our learning procedure. Overall cost: 79.2438

approximation of the current step.

Remark 7.6. We will use the common mathematical notation for the curvature sign. That is, a positive curvature means that the unit tangent vector rotates counter-clockwise as a function of the parameter along the curve.

For simplicity, we decided to approximate the motion of the finite horizon by the local geometry of the instantaneous position in the nominal trajectory. We expected that a sufficient control frequency would allow the system to properly track the general trajectory even if its local properties undertook steep temporal changes. For the pusher-slider system mentioned before, the base trajectory was chosen by numerically computing the curvature of the objective trajectory at every control step and using the base trajectory with the most similar curvature value.

We would then introduce dynamic families to the FOM problem. At each control step, the family used to track the trajectory would consist on the k best modes for the base trajectory that approximates the local geometry of the objective trajectory (together with the chameleon mode) to solve the MPC problem on that instant. The chameleon mode was shared between the dynamical families to ensure that the sequential consistency property was not lost.

The main motivation for this method was to reduce the online cost of the FOM approach at the expense of more offline computations. We expected that choosing the best family dynamically would reduce the number of modes per family required to obtain good results in a broad range of trajectories. Having n modes in a family implies solving **n** convex optimization problems at every control step, so we wanted to keep it as low as possible.

The learning algorithm was run on 10 pairs of curvatures (for each curvature k, we considered the complementary curvature $-k$) generated by setting the p_x parameter uniformly from $-\frac{3b}{4}$ to $\frac{3b}{4}$, where b is the length of the side of the slider. Using equation (TODO: add equation in previous section), the associated curvatures were obtained.

Generating general feasible trajectories for the controller to stabilize is a complex problem in itself. We could use MPC to compute an optimal path to a certain destination using the distance to the destination

as the cost function but this would have two important drawbacks. The first is that it would probably follow straight lines whenever possible, providing simple trajectories to stabilize. The second is that with MPC we can only obtain a discrete trajectory. We could approximate the obtained trajectory using splines, but we would not be able to guarantee feasibility of the trajectory defined by the state variable and the controller in the interpolated regions between control points.

The planning problem escaped the scope of this study and, in order to both simplify it and to check the robustness of the controller, we decided to create trajectories by simple concatenation of linear and circular trajectories. This simplified the planning problem and created punctual discontinuities in the position of the pusher (for the straight lines the pusher was constantly at the center of the slider side, and for the circular trajectories the pusher was constantly at the distance p_y associated to the circle and there were discontinuities at the changes between trajectories). We considered important to study whether the controller was capable to react to these temporally localized points of uncertainty were there was no feasible trajectory to immediately change from one trajectory to the next.

With this formulation we were able to successfully track the general trajectory and we were able to reduce the required finite horizon to track properly the trajectory to 15 steps, as can be seen in Figure **??**.

TODO: Wrap-up

Remark 7.7. The circular trajectories used in the simulations of the DFOM had curvatures different than the ones used for the learning protocol, so the DFOM algorithm had to determine the base circle which was more similar to the nominal trajectory one.

8. Discussion and Conclusion

1. MIQP takes too long, it's execution time depends on the geometry of the problem (different time at each iteration). The results may depend on the implementation chosen. If you don't have a software that allows you to tune everything it is too complicated to implement. Clustered MIQP is no better. We should try MIQP with tail (we expect good results but bad time)

9. Things to do if we had time

- Study modes correlation
- Study optimal mode dependency wrt initial condition.
- Change the uniformly distributed initial conditions by normally distributed.

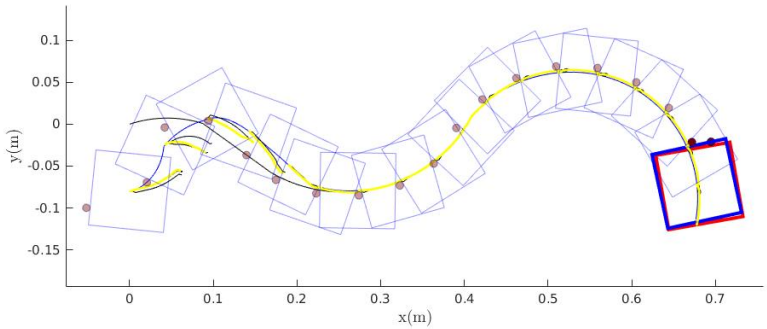

Figure 28: DFOM with 20 curvatures using the 4 best modes for each base trajectory. Finite horizon of 35 steps.

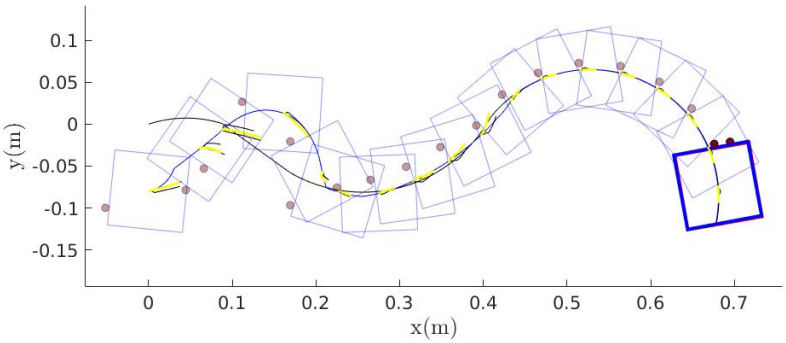

Figure 29: DFOM with 20 curvatures using the 4 best modes for each base trajectory. Finite horizon of 15 steps.

References

[1] Gurobi - mixed-integer programming (mip) - a primer on the basics. `https://www.gurobi.com/resources/getting-started/mip-basics`. Accessed: 2017-04-24.

[2] Gurobi optimizer description. `http://www.gurobi.com/products/gurobi-optimizer`. Accessed: 2017-04-24.

[3] G. Amontons. De la résistance causée dans les machines. *Mémoires de lAcadémie royale des Sciences*, pages 206–227, 1699.

[4] M. Anitescu and F. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(93):231–247, 1996.

[5] Dimitri P. Bertsekas, John N. Tsitsiklis, and Cynara. Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.

[6] Nikhil Chavan-Dafle and Alberto Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[7] R.W. Cottle, J.S. Pang, and R.E. Stone. The linear complementarity problem. *Academic Press*, 1992.

[8] C. A. Coulomb. Théorie des machines simples en ayant égard au frottement de leurs parties et à la roideur des cordages. *Mémoires de mathématique et de physique présentés à lAcadémie royale des Sciences*, 1781.

[9] Nima Fazeli, Roman Kolbert, Russ Tedrake, and Alberto Rodriguez. Parameter and contact force estimation of planar rigid-bodies undergoing frictional contact. *The International Journal of Robotics Research*, 2016.

[10] Nima Fazeli, Russ Tedrake, and Alberto Rodriguez. Identifiability analysis of planar rigid-body frictional contact. In *ISRR*, 2015.

[11] C. S. Gillmor. Coulomb and the evolution of physics and engineering in eighteenth-century france. *Princeton University Press*, 1971.

[12] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[13] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 2. dynamics of motion. *Wear*, 143(2):331–352, 1991.

[14] François Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *Workshop on Algorithmic Foundation of Robotics (WAFR)*, 2016.

[15] R.D. Howe, I. Kao, and M.R. Cutkosky. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. *International Conference on Robotics and Automation (ICRA)*, 1988.

[16] Roman Kolvert, Nikhil Chavan-Dafle, and Alberto Rodriguez. Experimental validation of contact dynamics for in-hand manipulation. *International Symposium on Experimental Robotics (ISER)*, 2015.

[17] Soo Hong Lee and M R Cutkosky. Fixture planning with friction. *Transactions of the ASME*, 113:320–327, 1991.

[18] K.M. Lynch, H. Maekawa, and K. Tanie. Manipulation and active sensing by pushing using tactile feedback. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:416–421, 1992.

[19] Matthew T Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man and Cybernetics*, 11(6):418–432, 1981.

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

[21] Matthew T Mason. Haptic perception, grasping and manipulation with simple robotic grippers, 2 2016. Remarks by Mathew T Mason at the CITRIS People and Robots Seminar Series, UC Berkeley [Accessed: 2017 04 20].

[22] E. Meyer. Nanoscience: Friction and rheology on the nanometer scale. *World Scientific*, page 7, 2002.

[23] H. Moseley. On the equilibrium of the arch. *Trans. Cambridge Philosophical Soc.*, pages 293–314, 1835.

[24] J.S. Pang and J.C. Trinkle. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. *Mathematical Programming*, 73(2):199–226, 1996.

[25] Alberto Rodriguez and Matthew Mason. Two finger caging: Squeezing and stretching. *Workshop on the Algorithmic Foundations of Robotics(WAFR)*, 2008.

[26] D. 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.

[27] C. Truesdell. Essays in the history of mechanics. *Springer-Verlag*, 1968.

[28] Wang Ya-Feng, Sun Fu-Chun, Zhang You-An, Liu Hua-Ping, and Min Haibo. Getting a suitable terminal cost and maximizing the terminal region for mpc. *Mathematical Problems in Engineering*, 2010.

A. Simulation Interface Code

You can include here an appendix with details that can not be included in the core of the document. You should reference the sections in this appendix in the core document.