2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

Flatness-based MPC using B-splines transcription with application to a Pusher-Slider System

Thomas Neve[1,2,*], Tom Lefebvre[1,2], Sander De Witte[1,2] and Guillaume Crevecoeur[1,2]

***Abstract*— This work discusses the use of model predictive control (MPC) for the manipulation of a pusher-slider system. In particular we leverage the differential flatness of the pusher-slider in combination with a B-splines transcription to address the computational demand that is typically associated to real-time implementation of an MPC controller. We demonstrate the flatness based B-spline MPC controller in simulation and compare it to a standard MPC implementation approach using direct multiple shooting. We evaluate the computational advantage of the flatness based MPC empirically and document computational acceleration up to 65%.**

I. Introduction

When we manipulate objects in a daily setting our actions reach further than just simple grasping and releasing. We perform an array of different tasks, such as pushing, pulling, sliding, and lifting. Therefore, we argue that pushing is a motion primitive of practical significance for robotic manipulation as well. Pushing extends the normal capabilities resulting in a wide variety of applications, such as the manipulation of hard to grasp objects due to for example their weight or shape, or positioning tasks [1]. Moreover, we contend that pushing an object allows to simplify the design of the end-effector without significantly compromising the maneuverability of the object. The pusher-slider system, discussed in this work, represents a basic non-prehensile manipulation task where the goal is to control the motion of the slider through the pusher. The system consists of a sliding object (the slider) and a single contact point (the pusher).

To achieve optimal control of a pusher-slider system, a precise model of the slider's behavior under pushing is required, as well as a robust control strategy to address model inaccuracies and external disturbances. Model Predictive Control (MPC) has proven to be an effective control strategy in complex systems due to its ability to anticipate future behavior and act optimally in changing environments [2]. MPC optimizes dynamic behavior over a prediction horizon at each sampling period while adhering to a set of constraints. Common approaches include direct methods, which transforms the optimal control problem into a numerical optimization problem [3]. Several techniques exist here to transcribe the control problem, typically using polynomial or piecewise constant parameterization. A common direct approach is Direct Multiple Shooting (DMS) which considers both the state and control as optimization variables. Subsequent states in the prediction horizon are tied together using equality constraints between each control interval [4].

Solving this optimization problem in real-time can be computationally demanding, limiting its practical application. In this work we attempt to adress this challenge by exploitation of the differentially flat properties of the dynamics model. Differential flatness is a property of a class of dynamical systems characterized by the ability to define all states and controls of the system as a set of specific differential variables and their derivatives. This property can be particularly useful for both solving trajectory planning [5] and tracking problems [6]–[8]. In [9] full optimal flatness based MPC has also been used for the control of underactuated surface vessels. Most work however does not solve the full optimal control control problem each MPC iteration. It is still unclear how the differential flatness property is best used in an MPC implementation and what the impact is on computational requirement.

We present an MPC controller that leverages the differentially flat properties of the introduced kinematic pusher-slider model and employs a B-spline to parameterize the trajectory. This significantly reduces the complexity of the trajectory optimization problem due to less optimization variables and constraints compared to the DMS approach. The B-spline is used to represent the trajectory of the flat coordinates of the pusher-slider system, after which the other variables can be deduced from the flat coordinates. However, the nonlinear flat expressions used to compute the original states and controls from the flat trajectory also add some nonlinearity to the objective and constraints. It is unclear how such a flatness based MPC would compare to standard transcription methods such as DMS. Our contribution consists of the application of a flatness based MPC strategy to a pusher-slider system, and comparing its performance in terms of computational requirements to the DMS approach.

II. Pusher-Slider Model

The pusher-slider system can be modelled as a quasi-static model where no accelerations occur. This model is physically

[1]D2Lab research group, Department of Electromechanical, Systems and Metal Engineering, Ghent University, Tech Lane Ghent Science Park 913, B-9052 Zwijnaarde, Belgium
[2]Core Lab MIRO, Flanders Make.
[*]Corresponding author: thomas.neve@ugent.be.

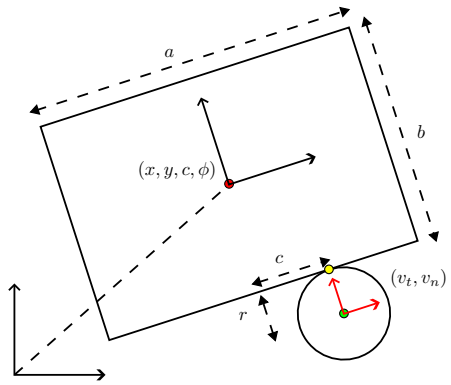


Fig. 1: *State and input values pusher-slider system.*

valid in the following cases.

1) The motions of the slider are slow enough that inertial forces are negligible compared to friction forces.
2) The friction forces at the contact point between pusher and slider are negligible with respect to the friction forces between the slider and the ground.

We start by describing the kinematic model and continue by discussing the differentially flat properties of the resulting model and the accompanying expressions of the pusher-slider system.

A. Kinematics

The system consists of a sliding object (the slider) and a single contact point (the pusher). We define the state of the system as

$$\mathbf{x}^\top = \begin{pmatrix} x & y & c & \phi \end{pmatrix}$$

and the input as

$$\mathbf{u}^\top = \begin{pmatrix} v_t & v_n \end{pmatrix}$$

with variables shown in figure 1. Here (x, y, ϕ) represents the planar configuration of the slider in the global frame of reference and c the relative position of the pusher contact with the slider. The inputs v_t and v_n denote, respectively, the speed tangent to and normal to the surface against which is pushed. Some additional geometric parameters are defined as well: with a and b denoting the length of the slider's side, and r denoting the radius of the pusher.

By assuming quasi-static interaction and frictionless contact between the pusher and the slider, the model becomes entirely kinematic. The former means physically that we assume that in any configuration of the slider the ground friction is large enough to keep the slider stationary if not acted upon. The following equations emerge

$$\begin{aligned}
\dot{x} &= -\frac{\beta^2}{\beta^2 + c^2} v_n \sin(\phi) \\
\dot{y} &= \frac{\beta^2}{\beta^2 + c^2} v_n \cos(\phi) \\
\dot{c} &= v_t - \left(\frac{b}{2} + r\right) \frac{c}{\beta^2 + c^2} v_n \\
\dot{\phi} &= \frac{c}{\beta^2 + c^2} v_n
\end{aligned}$$

where β^2 is a factor that depends on the geometry of the slider. In the case of a rectangular geometry $\beta^2 = \frac{1}{12}(a^2 + b^2)$. For the complete derivation we refer to the work by Lefebvre et al. [10].

B. Differential flatness

Differential flatness or just flatness is a generalisation of the notion of inverse dynamics for underactuated nonlinear systems, hence $n_x > n_u$ where n_x and n_u denote the state $\mathbf{x} \in \mathbb{R}^n$ and control $\mathbf{u} \in \mathbb{R}^m$ dimensionality. To circumvent the over-defined inverse dynamics problem, a set of flat coordinates $\mathbf{y} \in \mathbb{R}^{n_u}$ is selected with the same dimensionality as the control. The flat coordinates themselves may not have physical meaning.

$$\mathbf{y} = y(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, ..., \mathbf{u}^{(p)})$$

Then the system state and the control are expressed as a function of the flat coordinates and its derivatives.

$$\mathbf{x} = x(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, ..., \mathbf{y}^{(q)})$$

$$\mathbf{u} = u(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, ..., \mathbf{y}^{(q)})$$

It can be shown that the equations of motion of the pusher-slider are flat [10]. A set of flat coordinates with the same number of coordinates as the input $n_u = 2$ is defined. For the pusher-slider system the flat coordinates are taken as the Cartesian coordinates.

$$\mathbf{p}^\top = \begin{pmatrix} x & y \end{pmatrix}$$

The flat expressions for the control input and auxiliary states can be derived as a function of these coordinates and their derivatives.

$$\begin{aligned}
c &= \beta^2 \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}^3} \\
\phi &= -\arctan\frac{\dot{x}}{\dot{y}} \\
v_t &= \left(1 + \beta^2 \frac{(\ddot{x}\dot{y} - \dot{x}\ddot{y})^2}{(\dot{x}^2 + \dot{y}^2)^3}\right) \sqrt{\dot{x}^2 + \dot{y}^2} \\
v_n &= \beta^2 \frac{\dot{x}\dddot{y} - \dddot{x}\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}^3} + 3\beta^2 \frac{(\ddot{x}\dot{y} - \dot{x}\ddot{y})(\dot{x}\ddot{x}\dot{y} + \dot{y}\ddot{y})}{\sqrt{\dot{x}^2 + \dot{y}^2}^5} + \cdots \\
&\quad \left(\frac{b}{2} + r\right) \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}
\end{aligned} \tag{1}$$

III. Model Predictive Control

Model predictive control (MPC) is a control strategy that optimizes over a prediction horizon to steer a dynamic system in an optimal way whilst satisfying a set of constraints.

Consider the following optimal control problem (OCP)

$$\min_{\mathbf{u}(t)} l_T(\mathbf{x}(T)) + \int_t^T l(\mathbf{x}(t), \mathbf{u}(t))dt$$
$$\text{s.t.} \begin{cases} \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{x}(0) = \mathbf{x}_{\text{start}} \\ \mathbf{x}(T) = \mathbf{x}_{\text{goal}} \\ \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \leq \mathbf{0} \end{cases} \tag{2}$$

with $\mathbf{x}(0)$, $\mathbf{x}(T)$ and $\mathbf{h}$ the initial, final state and path constraints respectively. Each MPC iteration, the OCP is solved to acquire the optimal control signal $\mathbf{u}(t)$. This control signal could be applied in open-loop to the system to perform in an optimal manner. However, due to model inaccuracies or external disturbances the system could quickly diverge from the expected behavior. To account for this, the actual state following the applied control is used as the new initial state, and the optimization process is repeated. Effectively closing the loop. Typically, the control signal $\mathbf{u}(t)$ is considered over the same time horizon T on each iteration. Should one want to converge to a specific goal state within the given time horizon, it is also possible to reduce this horizon on each iteration. Thus modifying the OCP formulation on each step.

MPC could be used for both trajectory tracking and generation. The former uses an MPC controller to track a predefined trajectory [7], [8]. In this work however we focus on trajectory generation, generating a new trajectory at each time step. Ergo we reconsider the full optimal control problem every time step.

The common approach to solving the optimal control problem in MPC is through the use of direct methods, which translate the control problem into a numerical optimization problem. Within the class of direct methods several approaches exist to transcribe the OCP into a trajectory optimization problem. A popular approach for longer time horizons is direct multiple shooting (DMS), which considers both the system state and control as optimization variable. Another approach is the use of polynomial parameterization, e.g. B-splines [9], of the trajectory. This approach can be particularly usefull in combination with a differentially flat system. We continue this section with a more detailed discussion on both approaches. These techniques will then be applied and compared onto the pusher-slider system in terms of computational performance to highlight their respective advantages and disadvantages.

A. Direct Multiple Shooting

Direct multiple shooting (DMS) is a strategy to transcribe a trajectory optimization problem into a numerical optimization problem. The prediction horizon of the problem is split, using a discretized dynamics model, into several equidistant shooting nodes. The multiple shooting transcription of the control problem (2) results in the following nonlinear program (NLP)

$$\min_{\mathbf{u}_{0:N-1}, \mathbf{x}_{0:N}} \sum_{i=0}^{N-1} l(\mathbf{x}_i, \mathbf{u}_i)$$
$$\text{s.t.} \begin{cases} \mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i) \\ \mathbf{x}_0 = \mathbf{x}_{\text{start}} \\ \mathbf{x}_N = \mathbf{x}_{\text{goal}} \\ \mathbf{x}_{\min} \leq \mathbf{x}_i \leq \mathbf{x}_{\max} \\ \mathbf{u}_{\min} \leq \mathbf{u}_i \leq \mathbf{u}_{\max} \end{cases} \tag{3}$$

with N the number of discretization steps, l the cost function and $\mathbf{f}$ the discrete dynamics of the pusher-slider over one shooting interval. Note the addition of the state trajectory x_t as an optimization variable. This component is key to the mutliple shooting approach. Continuity in the state trajectory between shooting nodes is enforced through a set of continuity constraints using f.

The addition of the states x_t to the optimization might seem counterproductive compared to a single shooting approach where only the controls u_t are considered as optimization variables. But compared to single shooting, mutliple shooting is more flexible in initalizing the problem, and has improved convergence properties [4], [11].

B. Flatness based MPC

The continuity constraint of (3) is implicitly fulfilled in the flat parameterization (1) of the system. Put differently, the entire class of feasible state-action trajectories is encoded by the class of smooth flat paths. Therefore, the differential flatness properties of the pusher-slider could be leveraged to transcribe the control problem without the need for a set of constraints to enforce continuity. To parametize the state trajectory, B-splines can be used to represent the flat trajectory without considering the controls themselves as optimization variables directly. From this parameterized trajectory the full state and controls can still be inferred using (1).

1) B-splines: In this section we give an introduction to B-splines. For an extensive theory we refer to the related literature [12], [13].

B-splines, first introduced in [13], consist of a union of local curve segments which are each active on a specific interval. Its segmented nature allows for very efficient tailoring to desired local changes [14]. This property also makes it particularly useful for trajectory optimization where one might desire a local change without affecting global behaviour, which in the end will result in sparse Jacobian and Hessian structures of the nonlinear program.

Consider the expansion

$$S(\tau) = \sum_{i=0}^{n} N_{i,k} p_i, \quad \tau \in [0, T] \tag{4}$$

where the spline $S(\tau)$ is defined over the closed interval $[0, T]$ which is subdivided into m sub intervals with $m = k + n + 1$. The vector $\mathbf{p} = [p_0, \ldots, p_n]$ constitutes the set of control points which acts as a set of weights on the basis functions $N_{i,k}$ where k is the order of the B-spline, $n+1$ is the number

of control points and m defining the number of knots. The basis functions can be determined according to the following recursion formula [15].

$$N_{i,0}(\tau) = \begin{cases} 1 & \tau_i \leq \tau \leq \tau_{i+1} \\ 0 & \text{otherwise} \end{cases},$$
$$N_{i,j}(\tau) = \frac{\tau - \tau_i}{\tau_{i+j} - \tau_i} N_{i,j-1}(\tau) + \frac{\tau_{i+j+1} - \tau}{\tau_{i+j+1} - \tau_{i+1}} N_{i+1,j-1}(\tau) \tag{5}$$

It can be seen here that the closed interval $[0, T]$ is divided uniformly through a knot vector.

$$\boldsymbol{\tau} = [\tau_0, \tau_1, \ldots, \tau_m] \tag{6}$$

At the knot points, the polynomials are joined and connected in a continuous manner. From the recursion (5) it is clear that each basis function $N_{i,k}$, with weight p_i, is only active on a subset of the interval $[0, T]$. More specifically, it is nonzero on the interval $[\tau_i, \tau_{i+k+1})$, resulting in the B-spline not being defined at the start and the ending of the interval $[0, T]$. Choosing the knot vector with duplicate knots at the ends as

$$\boldsymbol{\tau} = [\underbrace{0 \ldots 0}_{\text{k}} \ \underbrace{0 \ldots T}_{\text{internal knots}} \ \underbrace{T \ldots T}_{\text{k}}], \tag{7}$$

alleviates this and also clamps the start and endpoint of the spline to the two end control points, $S(0) = p_0$ and $S(T) = p_n$. This is also often referred to as a clamped uniform B-spline.

The derivative of a B-spline is simply a function of B-splines of a lower degree. Since a B-spline function of order k consists of polynomials of order $n - 1$, it's derivatives are continuous up to the derivative of degree $n - 2$. The flat expressions require a derivative of the flat coordinates up to order q. Thus the order of the B-spline used to represent the flat trajectory should be at least of order $q+2$. This is usually chosen as the minimum value to avoid numerical issues [9].

2) B-spline transcription: We can now represent our OCP as a numerical optimization problem using B-spline transcription. Here we consider the control points $\mathbf{p}_{0:\text{n}}$ as optimization variables that bend the continuous B-spline curve according to some objective and constraints. With an equidistant knot vector, resulting in a uniform B-spline, we can evaluate the B-spline function at a set of collocation $N + 1$ points. Using the flat expressions (1) we can express the objective and constraints at the collocation points, resulting in an optimization problem without the need for continuity constraints.

$$\min_{\mathbf{p}_{0:\text{n}}} \sum_{i=0}^{N} l(\mathbf{x}_i(\mathbf{p}_{0:\text{n}}), \mathbf{u}_i(\mathbf{p}_{0:\text{n}}))$$
$$\text{s.t.} \begin{cases} \mathbf{x}_0(\mathbf{p}_{0:\text{n}}) = \mathbf{x}_{\text{start}} \\ \mathbf{x}_N(\mathbf{p}_{0:\text{n}}) = \mathbf{x}_{\text{goal}} \\ \mathbf{x}_{\text{min}} \leq \mathbf{x}_i(\mathbf{p}_{0:\text{n}}) \leq \mathbf{x}_{\text{max}} \\ \mathbf{u}_{\text{min}} \leq \mathbf{u}_i(\mathbf{p}_{0:\text{n}}) \leq \mathbf{u}_{\text{max}} \end{cases} \tag{8}$$

IV. Simulation Experiments

We will now verify the computational characteristics of the proposed transcription methods in function of real-time appliction of the corresponding MPC approach. Therefore we define several validation cases. Due to the arbitrariness of the global frame of reference we can consider initial position $[x, y] = [0, 0]$ and orientation $\phi = 0$ without loss of generality. Each case the MPC controller is used to control the pusher to manoeuvre the slider according to some objective towards the goal location and orientation. Each MPC iteration an OCP is solved at discrete time points with a fixed time step $\Delta t = \frac{T}{N+1}$. Two different transcription techniques, DMS and B-spline transcription, are used to build the MPC controller. For each transcription technique we consider both a fixed and shrinking horizon version, resulting in four different implementations. The controllers are implemented in CasADi [16] and solved using IPOPT [17]. In the simulation, noise was added to each control, $u_{noise} \sim \mathcal{N}(0, \Sigma)$, with $\Sigma = \text{diag}(0.2, 0.2)$.

More concretely, the OCP takes the following generic form for both the spline and DMS transcription.

$$J = \min_{\mathbf{u}_{0:N-1}} \sum_{i=0}^{N-1} \mathbf{u}_i^T R \mathbf{u}_i + \mathbf{x}_i^T Q \mathbf{x}_i + \mathbf{x}_N^T P \mathbf{x}_N$$
$$\text{s.t.} \begin{cases} \mathbf{x}_0 = \mathbf{x}_{\text{start}} \\ -0.4 \leq c \leq 0.4 \end{cases} \tag{9}$$

Next to the objective desribed above, we add a regularization term to the B-spline transcription case, $J_{\text{tot}} = J + w_{\text{reg}} J_{\text{reg}}$.

$$J_{\text{reg}} = \sum_{i=0}^{n-1} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|^2 \tag{10}$$

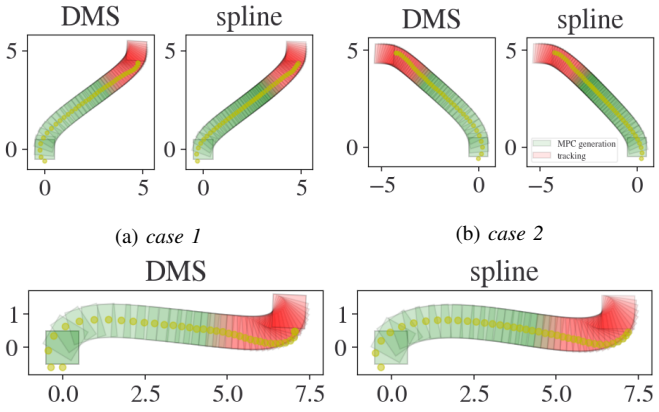
This term was added with the aim of maintaining a uniform distance between the control points. We have found empirically that this can improves numerical stability significantly.

Initial results here were promising for the most part. However, once the slider got close to the goal, MPC performance starts deteriorating in the form of increasing computation time and numerical instability. This phenomenon ocurred for both the B-spline and DMS approach but was more outspoken for the flatness based MPC which starts to fail slightly sooner. For this reason, the MPC controller is used up untill a switching point where the final number of steps are handled by a tracking controller. The tracking controller uses a DMS formulation to track the final solution from the MPC generation.

We start by describing our methodology of an MPC with typical fixed preview horizon.

A. Fixed horizon

Each MPC step an OCP with preview horizon $T = 1$ and $N = 20$ discretization steps is solved. The first control from the solution, u_0, is applied to the system and the aformentioned OCP is resolved with the new state, position and orientation, of the pusher-slider.



(a) *case 1* (b) *case 2*

(c) *case 3*

Fig. 2: *All cases with a fixed horizon MPC. The settings are reported in table I.*

settings	diag(R)	diag(Q)	diag(P)	w_{reg}
case 1	$(2, 2)$	$(1, 1, 10, 0)$	$(500, 500, 2000, 0)$	2
case 2	$(2, 2)$	$(0.5, 0.5, 0, 0)$	$(500, 500, 2000, 0)$	5
case 3	$(0.5, 0.5)$	$(1, 1, 0, 0)$	$(500, 500, 2000, 0)$	10

TABLE I: *Settings used for the fixed horizon MPC. For the cost matrices, the diagonal components are reported.*

1) Implementation details: The MPC controller is used up untill the slider starts getting close to the goal, $\|(x, y) - (x_{\text{goal}}, y_{\text{goal}})\| = 2.5$. Afterwards the tracking controller is activated.

The solution resulting from the B-spline transcription is a continous trajectory in τ. The required control can be easily computed with the expressions (1) using the flat coordinates and its derivatives evaluated at $\tau = 0$.

In our experiments we use a B-spline of order $k = 5$ with number of knots $m = 12$ resulting in $n = 6$ control points.

2) Discussion: The results for three test cases are shown in figure 2a, 2b and 2c with the asociated computation times indicated in figure 3. The MPC phase is indicated in green and tracking phase of the control is indicated in red. The indicated computation times are only given for the MPC phase.

In all cases, the B-spline transcription with flat trajectories outperforms the DMS method in terms of computation time. The main reason for this is the substantially lower amount of optimization variables in the B-spline transcription. Only n control points are used, resulting in $n \cdot n_u$ optimization variables. In contrast, the DMS approach requires a control and state vector for each of the N shooting nodes, resulting in $N \cdot (n_x + n_u) + n_x$ variables.

B. Shrinking horizon

Instead of keeping the horizon fixed, each subsequent step the horizon is reduced with the goal of converging to the goal within the intended number of steps $N = 20$ and time horizon $T = 1$. In other words, after each control step, the preview horizon T is reduced by Δt.

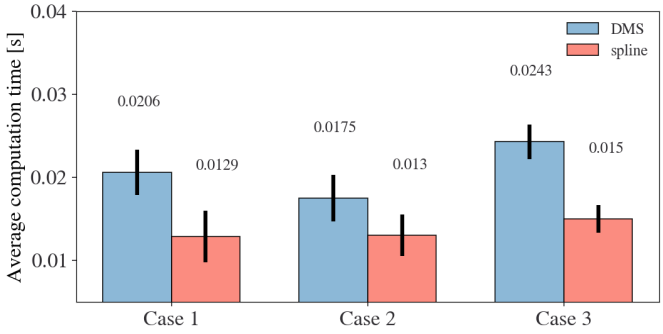


Fig. 3: *Computation time of an MPC controller with fixed preview horizon for all cases.*

settings	diag(R)	w_{reg}
case 1	$(0.1, 0.1)$	1
case 2	$(0.5, 0.5)$	1
case 3	$(0.4, 0.4)$	10

TABLE II: *Settings used for the shrinking horizon MPC. For the cost matrices, the diagonal components are reported.*

1) Implementation details: In contrast to the fixed horizon formulation, we also add a terminal constraint to the problem, $\mathbf{x}_N = \mathbf{x}_{\text{goal}}$. This change in the formulation also allows us to drop the Mayer term P and Q from the lagrangian term in (9). Relying solely on the terminal constraint and reducing horizon to converge to the goal. The tracking controller is also activated after $N_{\text{MPC}} = 15$ steps.

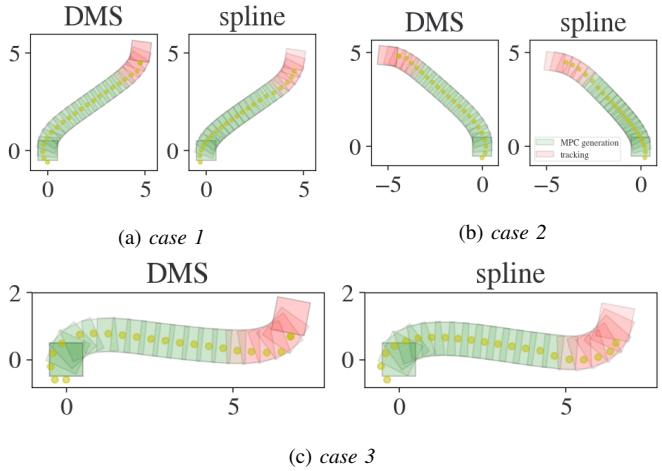
Since the B-spline transcription results in a continuous flat geometric trajectory, redefining the time parameterization is very practical. One only needs to replace the time horizon T in the B-spline parameterization (4) with the new reduced horizon. The number of optimization variables also remains identical throughout the manoeuvre.

For the DMS approach, a reducing horizon is conceptually relatively simple but can take a bit more effort in implementation. Each control step the number of steps in the preview horizon N is reduced by 1 in order to retain the same time step Δt under a reducing time horizon. This also results in a reducing number of variables and constraints on each subsequent iteration. Here we opted to rebuild the underlying NLP on each iteration.

2) Discussion: The results for three test cases are shown in figure 4a, 4b and 4c with the asociated computation times indicated in figure 5. Similar to the fixed horizon scenario, the B-spline method outperforms the DMS method in computation times. Notice the higher variance in the computation time of the DMS approach. This is unsurprising as the number of optimization variables reduces when the number of horizon steps is reduced, resulting in a reduced computational demand upon nearing the goal.

V. Conclusion

In this work we demonstrated the use of a flatness based MPC using a B-splines transcription to parameterize a flat trajectory on a pusher-slider system. We compared this MPC



(a) *case 1*

(b) *case 2*

(c) *case 3*

Fig. 4: *All cases with a shrinking horizon horizon MPC. The settings are reported in table II.*

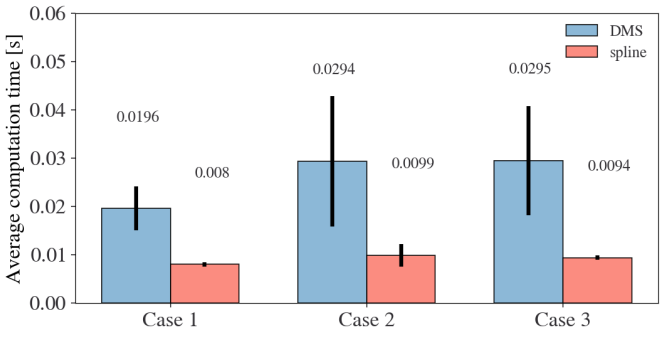


Fig. 5: *Computation time of an MPC controller with shrinking horizon for all cases.*

methodology to a typical MPC controller using DMS and evaluated how both methodologies compare on several cases where a slider manoeuvre towards a goal is performed. We did this for both a fixed and shrinking horizon implementation.

Results show that there is benefit in exploiting the differential flatness of the pusher-slider in combination with B-splines. One of it's advantages includes a reduced amount of optimization variables required to transcribe the optimal control problem to an optimization problem. Furthermore, the continuous parameterization of the flat trajectory offers a more elegant way to implement a shrinking horizon MPC.

Acknowledgment

This work was supported by the Flanders Make projects DIRAC and the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

References

[1] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 30–37. IEEE, 2016.

[2] Francois Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. CoRR, abs/1611.08268, 2016.

[3] Moritz Diehl, Hans Joachim Ferreau, Niels Haverbeke, L Magni, DM Raimondo, and F Allgower. Efficient numerical methods for nonlinear mpc and moving horizon estimation, 2009-01-01.

[4] Jan Albersmeyer and Moritz Diehl. The lifted newton method and its application in optimization. SIAM J. on Optimization, 20(3):1655–1684, jan 2010.

[5] Florin Stoican, Ionela Prodan, and Dan Popescu. Flat trajectory generation for way-points relaxations and obstacle avoidance. In 2015 23rd Mediterranean Conference on Control and Automation (MED), pages 695–700. IEEE, 2015.

[6] Ngoc Thinh Nguyen, Ionela Prodan, Florin Stoican, and Laurent Lefèvre. Reliable nonlinear control for quadcopter trajectory tracking through differential flatness. IFAC-PapersOnLine, 50(1):6971–6976, 2017. 20th IFAC World Congress.

[7] Melissa Greeff and Angela P. Schoellig. Flatness-based model predictive control for quadrotor trajectory tracking. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6740–6745, 2018.

[8] Zejiang Wang, Jingqiang Zha, and Junmin Wang. Flatness-based model predictive control for autonomous vehicle trajectory tracking. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 4146–4151, 2019.

[9] Simon Helling, Max Lutz, and Thomas Meurer. Flatness-based mpc for underactuated surface vessels in confined areas. IFAC-PapersOnLine, 53(2):14686–14691, 2020. 21st IFAC World Congress.

[10] Tom Lefebvre, Sander De Witte, Thomas Neve, and Guillaume Crevecoeur. Differential Flatness of Slider–Pusher Systems for Constrained Time Optimal Collision Free Path Planning. Journal of Dynamic Systems, Measurement, and Control, 145(6), 04 2023. 061001.

[11] Rien Quirynen, Milan Vukov, and Moritz Diehl. Multiple Shooting in a Microsecond, pages 183–201. 01 2015.

[12] Carl De Boor. A practical guide to splines, volume 27.

[13] Contributions to the problem of approximation of equidistant data by analytic functions. Quarterly of Applied Mathematics, 4:112–141, 1946.

[14] Boris Rohal'-Ilkiv, Martin Gulan, and Peter Minarčík. Implementation of continuous-time mpc using b-spline functions. In 2019 22nd International Conference on Process Control (PC19), pages 222–227, 2019.

[15] Carl de Boor. On calculating with b-splines. Journal of Approximation Theory, 6(1):50–62, 1972.

[16] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. Mathematical Programming Computation, 11(1):1–36, 2019.

[17] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program., 106(1):25–57, mar 2006.