Receding-Constraint and Parallel-Constraint Model Predictive Control: novel methods to enhance Safety in Nonlinear Control

Elias Fontanari[1], Gianni Lunardi[1], Matteo Saveriano[1] and Andrea Del Prete[1]

***Abstract*— Ensuring constraint satisfaction is a key requirement for safety-critical systems, which include most robotic platforms. For example, constraints can be used for modeling joint position/velocity/torque limits, collision avoidance and to specify desired behaviors during operations. Constrained systems are often controlled using Model Predictive Control, because of its ability to naturally handle constraints, relying on numerical optimization. However, ensuring constraint satisfaction is challenging for nonlinear systems/constraints. A well-known tool to make controllers safe is the so-called *safe set* (a.k.a. control-invariant set). Unfortunately, for most nonlinear systems, as robot manipulators, safe sets cannot be exactly computed, but can only be approximated with numerical methods. This extended abstract presents two novel Model Predictive Control schemes that can guarantee safety under weaker requirements on the safe set. These methods replace the classic terminal constraint with novel receding constraints, leading to a higher number of completed tasks, while retaining safety guarantees. From a theoretical point of view these controllers are superior, in the sense that the safe set has to satisfy less strict conditions to ensure Recursive Feasibility. Moreover, they can rely on a safe task-abortion strategy to drive the system to an equilibrium state when a risk of constraint violation is detected. We evaluated our approaches on a simulated robot manipulator, empirically demonstrating their superiority to state-of-the-art MPC schemes.**

I. Introduction

Guaranteeing safety is a fundamental requirement in almost all robotics applications. Safety is typically formulated via a set of equality/inequality constraints that the system should satisfy at all times. For instance, such constraints can model joint limits, actuation bounds, collision avoidance, or even more abstract and intangible requirements that are difficult to describe using the traditional model-based state space. These kinds of constraints can be employed in a large variety of tasks that are difficult to formulate analytically, and can rely on uncertain safety specifications, such as natural language, semantic concepts, and high-dimensional sensor data (e.g., images).

Regardless of the type of constraint, ensuring persistent satisfaction is extremely challenging. This is the case for both recent data-driven approaches, often relying on Reinforcement Learning (RL) algorithms, and for model-based control methods such as Model Predictive Control (MPC). Indeed, if the system dynamics is nonlinear, MPC methods in general cannot easily guarantee safety.

The most common approach for guaranteeing safety relies on the knowledge of a so-called *safe set* (a.k.a. control-invariant set) [1], [2], or, equivalently, a Control Barrier Function (CBF) [3], [4]. However, computing exact safe sets for nonlinear systems is generally untractable. Hence, professionals have relied on numerical methods that can compute approximations of such sets/functions [5]–[12]. Unfortunately, safety guarantees are compromised if the safe set is not exact.

We present novel MPC schemes, namely Receding-Constraint MPC and Parallel-Constraint MPC, that ensure: i) safety, assuming the safe set is a *conservative* approximation of a specific backward reachable set; ii) recursive feasibility, assuming the safe set is N-step control invariant, which is a weaker assumption than classic control invariance. We compared our approaches with classic MPC schemes. Our methods, tested in simulation on a $Z1$ manipulator with 4 actuated joints, could successfully avoid constraint violation in more tests than classic MPC formulations, without compromising task performance.

Even if the presented frameworks have been only tested with traditional constraint specifications (i.e., collision avoidance), they can potentially be applied to intangible constraints, once such constraints, are used to compute a safe set, as in [13].

II. Background

A. Notation

- $\mathbb{N}$ denotes the set of natural numbers;
- $\{x_i\}_0^N$ denotes a discrete-time trajectory given by the sequence $(x_0, \dots, x_N)$;
- $x_{i|k}$ denotes the state at time step $k+i$ predicted when solving the MPC problem at time step k;

B. Problem statement

Let us consider a discrete-time dynamical system with state and control constraints:

$$x_{i+1} = f(x_i, u_i), \qquad x \in \mathcal{X}, \qquad u \in \mathcal{U}. \tag{1}$$

Our goal is to design a control algorithm to ensure *safety* (i.e., constraint satisfaction), while preserving performance (i.e., cost minimization) as much as possible. Let us define $\mathcal{S}$ as the set containing all the equilibrium states of our system:

$$\mathcal{S} = \{x \in \mathcal{X} \mid \exists\, u \in \mathcal{U} : x = f(x, u)\}. \tag{2}$$

To ensure safety, we will rely on the *M-Step Backward-Reachable Set* [1] of $\mathcal{S}$, which we denote as $\mathcal{V}_M$. Mathematically, it is defined as the subset of $\mathcal{X}$ starting from which it

[1]The authors are with the Industrial Engineering Department, University of Trento, Via Sommarive 11, 38123, Trento, Italy. {name.surname}@unitn.it

is possible to reach $\mathcal{S}$ in M steps:

$$\mathcal{V}_M \triangleq \{x_0 \in \mathcal{X} \,|\, \exists \{u_i\}_0^{M-1} : x_M \in \mathcal{S}, x_i \in \mathcal{X}, \\ u_i \in \mathcal{U}, \forall\, i = 0, \dots, M-1\}. \tag{3}$$

Being a backward reachable set of equilibrium states, the set $\mathcal{V}_M$ is control-invariant [1]. Namely, starting inside $\mathcal{V}_M$, we can remain inside $\mathcal{V}_M$ forever. Knowledge of the set $\mathcal{V}_M$ would make it easy to design a safe controller. However, we cannot assume to know $\mathcal{V}_M$ in general, because its computation can be extremely complex. We rely instead on the following, more realistic, assumption.

Assumption 1. *We know a conservative approximation of the set $\mathcal{V}_M$:*

$$\hat{\mathcal{V}}_M \subseteq \mathcal{V}_M \tag{4}$$

Note that $\hat{\mathcal{V}}_M$ need not be control invariant in general.

As discussed above, different methods exist to compute numerical approximations of $\mathcal{V}_M$. We have chosen to use a slightly modified version of the Viability-Boundary Optimal Control (VBOC) method [12]. The resulting approximation of $\mathcal{V}_M$ can be easily made conservative by introducing a user-defined safety margin to "shrink" the set. Now we discuss different approaches to exploit $\hat{\mathcal{V}}_M$ in an MPC formulation to try to achieve safety.

C. Model Predictive Control and Recursive Feasibility

Let us consider the following MPC problem:

$$\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \tag{5a}$$

$$\text{subject to} \quad x_0 = x_{init} \tag{5b}$$

$$x_{i+1} = f(x_i, u_i) \qquad i = 0 \dots N-1 \tag{5c}$$

$$x_i \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \dots N-1 \tag{5d}$$

$$x_N \in \mathcal{X}_N, \tag{5e}$$

where $\ell(\cdot)/\ell_N(\cdot)$ is the running/terminal cost, x_{init} is the current state, and $\mathcal{X}_N \subseteq \mathcal{X}$ is the terminal set [14].

Even though MPC is one of the most suited frameworks for controlling constrained systems, ensuring safety (i.e., constraint satisfaction) remains challenging when dynamics and/or constraints are nonlinear. The most common approach is based on Recursive Feasibility (RF), which guarantees that, if an MPC problem is feasible at the first loop, it will remain feasible forever. RF is guaranteed if the terminal set $\mathcal{X}_N$ is *control-invariant*, such as $\mathcal{V}_M$. Unfortunately, we do not know $\mathcal{V}_M$, but only $\hat{\mathcal{V}}_M$, which is not control invariant in general. Therefore, using $\hat{\mathcal{V}}_M$ as terminal set does not ensure RF: this means that our problem could become unfeasible, losing safety.

III. Safe Model Predictive Control

This section describes our novel Safe MPC schemes.

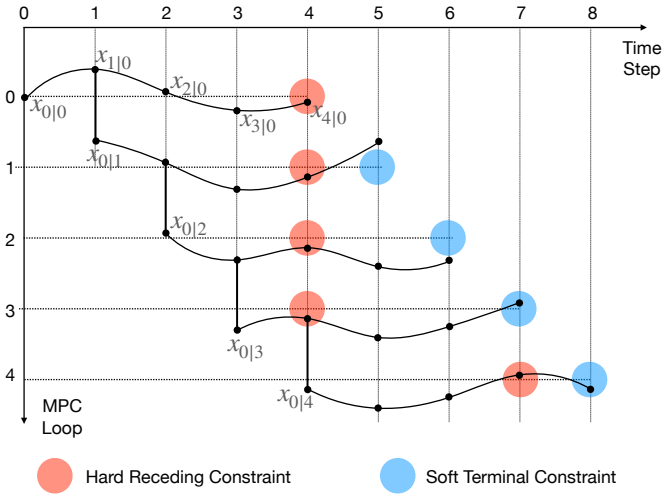

Fig. 1. Example of Receding-Constraint MPC with $N = 4$. After the MPC loop 3, the receding constraint slides forward because $x_{4|3} \in \hat{\mathcal{V}}_M$.

A. Safe Task Abortion

Our key idea to ensure safety relies on Assumption 1 and on the following two assumptions.

Assumption 2. *We have access to two computational units, which we refer to as unit A and unit B.*

Assumption 3. *We can solve the following Optimal Control Problem (OCP) for any $x_{init} \in \hat{\mathcal{V}}_M$, in $L+1$ time steps (with $0 \leq L \leq N-1$):*

$$\underset{\{x_i\}_0^M, \{u_i\}_0^{M-1}}{\text{minimize}} \quad \sum_{i=0}^{M-1} \ell_i(x_i, u_i) + \ell_M(x_M) \\ \text{subject to} \quad \text{(5b), (5c), (5d)}, \; x_M = x_{M-1} \tag{6}$$

Note that the only role of the cost function is to facilitate convergence.

OCP (6) finds a trajectory reaching an equilibrium state from x_{init}. Suppose we trigger the safe abort procedure at MPC loop k. The procedure consists of these steps:

1) unit A uses the MPC solution computed at loop $k-1$ to reach the safe state $x_{L+1|k-1} \in \hat{\mathcal{V}}_M$;
2) in parallel, unit B solves OCP (6), using $x_{L+1|k-1}$ as initial state;
3) after reaching $x_{L+1|k-1}$, we follow the solution of OCP (6) to reach an equilibrium state.

The hyper-parameter L should be set by the user as small as possible, while satisfying Assumption 3.

B. Receding Constraint MPC

As discussed in [15], our idea relies on the fact that, as long as at least one state $x_j \in \hat{\mathcal{V}}_M$ (with $1 \leq j \leq N$), we know that $x_1 \in \mathcal{V}_M$ because from x_1 we can reach x_j. Based on this insight, we suggest to adapt online the time step at which we constrain the state in $\hat{\mathcal{V}}_M$. If at the MPC loop $k-1$ we had $x_{j|k-1} \in \hat{\mathcal{V}}_M$, at the loop k we know that it is possible to have $x_{j-1|k} \in \hat{\mathcal{V}}_M$. This is sufficient to ensure

safety for j loops, during which this *receding constraint* would slide backward along the horizon. However, once the receding constraint reaches time step 0, it can no longer ensure safety. Therefore, we suggest to maintain also a soft constraint for the terminal state to be in $\hat{\mathcal{V}}_M$ and, after solving the MPC at loop $k-1$, to check whether $x_{N|k-1} \in \hat{\mathcal{V}}_M$; if that is the case, at loop k we can move the receding constraint forward on $x_{N-1|k}$, which ensures safety for other $N-1$ loops. An example is depicted in Fig. 1.

This method is better than the classic terminal-constraint approach because i) it ensures *safety* (if combined with the task abortion strategy), and ii) it ensures *recursive feasibility* for some MPC loops (i.e. j MPC loops, whenever a predicted state x_j is in $\hat{\mathcal{V}}_M$).

More precisely, in absence of stochastic effects, even at a theoretical level the Receding-Constraint MPC ensures *recursive feasibility* under a weaker assumption on the safe set than the terminal constraint approach, as outlined below.

Definition A set $\mathcal{A} \subseteq \mathcal{X}$ is *N-step control invariant* if, starting from any state in $\mathcal{A}$, it is possible to come back to it in exactly N time steps:

$$\forall x_0 \in \mathcal{A}, \exists \{u_i\}_0^{N-1} : \\ x_N \in \mathcal{A},\, x_i \in \mathcal{X},\, u_i \in \mathcal{U},\, \forall i = 0, \dots, N-1 \tag{7}$$

This is an extension of the well-known control invariance, with 1-step control invariance being equivalent to classic control invariance. So, assuming $\mathcal{V}_M$ is K-step control invariant, recursive feasibility is ensured if the horizon of the MPC N is equal to or greater than K, because at worst when $x_{0|k} \in \mathcal{V}_M$, it is possible to satisfy $x_{N|k} \in \mathcal{V}_M$, and the receding constraint can be moved forward.

C. Parallel-Constraint MPC

Instead of using a single state along the horizon to ensure safety, we could try to reach the safe set at different time steps exploiting parallel computation [16]. As long as at least one state $x_p \in \hat{\mathcal{V}}_M$ (with $1 \leq p \leq N$), we can ensure constraint satisfaction because $x_1 \in \mathcal{V}_{M+p-1}$. This is because from x_1 we can reach x_p in $p-1$ steps. Ideally, we would like to include the following constraint in our problem formulation:

$$(x_1 \in \hat{\mathcal{V}}_M) \vee (x_2 \in \hat{\mathcal{V}}_M) \vee \dots \vee (x_N \in \hat{\mathcal{V}}_M). \tag{8}$$

However, this type of constraint (OR constraints) is extremely hard to deal with for numerical solvers.

If we have access to N computational units, we can solve N problems in parallel, each constraining the state inside $\hat{\mathcal{V}}_M$ at a different time step $p \in [1, N]$:

$$\begin{aligned} \underset{X,U}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \\ \text{subject to} \quad & \text{(5b), (5c), (5d)}, x_p \in \hat{\mathcal{V}}_M \end{aligned} \tag{9}$$

Then, we can use the solution of one of the problems that have been successfully solved. Since our main concern is safety, we decide to use the solution satisfying the safe-set constraint at the furthest time. A scheme of the Parallel Constraint MPC is depicted in Fig. 2.

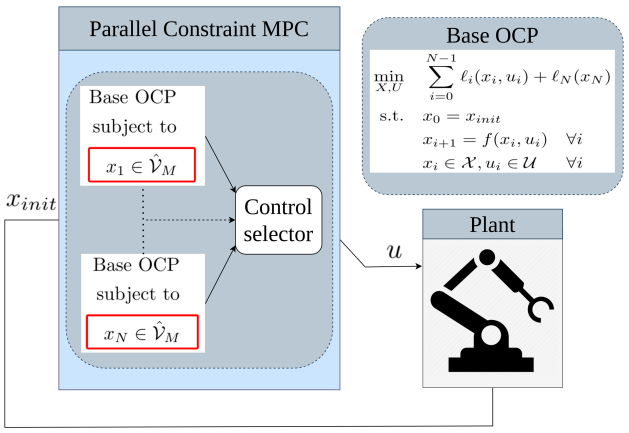

Fig. 2. Parallel MPC scheme. At each control step, N different problems with form (9) are solved in parallel, then the best solution in term of safety is chosen and applied to the plant.

IV. Simulation Results

A. Simulation Setup

To thoroughly evaluate the *safe abort* and the novel MPC formulations, we compared in simulation six MPC formulations of increasing complexity[1]:

- *Naive*: a classic formulation without terminal constraint, i.e., problem (5) with $\mathcal{X}_N = \mathcal{X}$. This is the baseline for all the experiments.
- *Zerovel*: a classic formulation that constrains the last state to be motionless. Basically, it uses the set of equilibria $\mathcal{S}$ as terminal set.
- *Soft Terminal* (*ST*): it introduces a soft terminal constraint set $\mathcal{X}_N = \hat{\mathcal{V}}_M$, as a first step towards RF.
- *Hard Terminal With Abort* (*HTWA*): classical formulation with hard terminal constraint. If the latter is not satisfied the last solution is shifted and used. It triggers the *safe abort* whenever for $N-L-1$ consecutive times the terminal constraint is violated.
- *Receding*: our novel formulation which uses a hard constraint for $x_r \in \hat{\mathcal{V}}_M$ and a soft constraint for $x_N \in \hat{\mathcal{V}}_M$. It triggers the *safe abort* if the receding constraint reaches the node $L+1$.
- *Parallel*: the novel formulation that solves N different problems simultaneously, in which the safe-set hard constraint is placed at different nodes and *safe abort* works as in the case of *Receding*.

For the simulations, we have used a 4-joint version of the Z1 manipulator. Starting from random configurations, the task is to reach a Cartesian position with the end-effector in an environment with fixed obstacles. Given $x = (q, \dot{q})$, the collision avoidance constraints can be formulated with

[1]Our open-source code is freely available at `https://github.com/idra-lab/safe-mpc`.

simple distance expressions $g(q)$. The running cost penalizes deviations from the Cartesian target p^{ref} and control efforts:

$$\begin{aligned} l(x,u) &= ||p(q) - p^{\text{ref}}||^2_Q + ||u||^2_R \\ Q &= 10^3 I_3, \quad R = 10^{-3} I_4, \end{aligned} \tag{10}$$

where $p(q)$ is the forward kinematics function to compute the end-effector position and I_h is the identity matrix with size h. $\hat{\mathcal{V}}_M$ was approximated using VBOC [12] and membership to the set is verified with the constraint:

$$(1-\alpha)\phi(x) - ||\dot{q}|| \geq 0, \tag{11}$$

where $\phi(\cdot)$ is a Neural Network (NN) computing an upper bound on the joint velocity norm [12], and $\alpha \in [0,1]$ is a user-defined safety factor introduced to ensure that $\hat{\mathcal{V}}_M \subseteq \mathcal{V}_M$.

We have run 300 simulations for each MPC formulation and each control horizon, starting from the same static random joint positions q_0, with time step $dt = 5\,\text{ms}$, safety margin $\alpha = 10\%$ and $L = 0$, assuming the OCP (6) can be solved in one step. CASADI [17] has been used as symbolic framework for the dynamics, costs, and constraints, while ACADOS [18] as OCPs solver and dynamics' integrator. To integrate the PyTorch [19] neural model representing the safe set inside ACADOS, we used L4CASADI [20]. The solver was used in RTI mode [21], to comply with the real-time constraints.

B. Constraint Violations

Figure 3 reports the percentage of constraint violations for each controller and for different horizons, using a safety margin $\alpha = 10\%$. As expected, *Zerovel* never failed, since it is the only controller employing an exact control invariant set, even if very small, causing a slow reaching of the target. In terms of safety, *Naive* and *ST* failed the most. *HTWA* performed better than the latter, thanks to the possibility of aborting the task when it detects potential failures. However, for control horizons larger than 25, excluding *Zerovel*, *Receding* and *Parallel* performed similarly and better than the others, while for shorter horizons *Parallel* was the best controller, failing only 7% of the tasks even with a very short horizon of 20 steps.

C. Costs

In terms of cost, Fig. 4 shows the average cost increment, considering only the tasks completed without violations by all controllers, with respect to the costs of the optimal solutions obtained solving an OCP with a horizon as long as the task duration. *Zerovel*, the safest controller, performed the worst. Surprisingly, *Naive* achieved worse results than the remaining controllers. All the other controllers performed very similarly, with a natural deterioration of performance for decreasing horizons.

V. CONCLUSIONS

We have presented two novel MPC formulations. Receding-Constraint MPC provides recursive feasibility guarantees under a weaker assumption on the used safe set

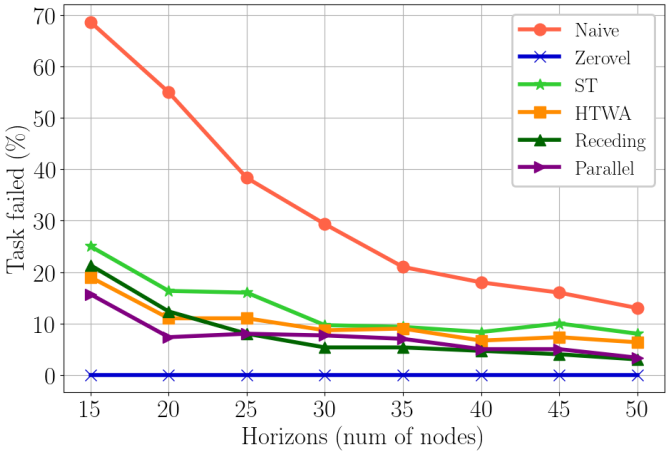

Fig. 3. Number of failures (constraint violations) for different lengths of the MPC horizon.

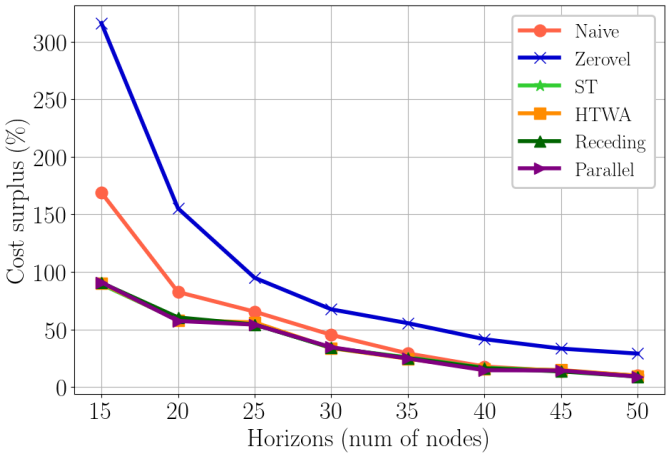

Fig. 4. Mean cost surplus w.r.t. the optimal costs computed solving a full-length OCP for each task not failed by any controller, for different lengths of the MPC horizon.

with respect to classic approaches. Parallel-Constraint MPC exploits parallel computation to improve safety. We have also presented a task-abortion strategy that allows to reach an equilibrium state whenever a risk of constraint violation is detected. Our results show the improved safety of our approaches w.r.t. state-of-the-art methods.

While our methods rely on weaker assumptions than standard approaches, these assumptions are still hard to verify, which can lead to constraint violations, as shown in our tests. Inspired by recent work [22], we want to investigate the use of robust optimization techniques to certify N-Step Control Invariance.

We also plan to account for uncertainties in the dynamics using robust optimization. While this work focused on model-based control methods, our approach could be extended in the future to *safety filters* based on intangible specification or used to make safe black-box policies as those obtained from RL.

REFERENCES

[1] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, pp. 1747–1767, 1999.

[2] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2017.

[3] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[4] Z. Wu, F. Albalawi, Z. Zhang, J. Zhang, H. Durand, and P. D. Christofides, "Control Lyapunov-Barrier function-based model predictive control of nonlinear systems," *Automatica*, vol. 109, p. 108508, 2019. [Online]. Available: https://doi.org/10.1016/j.automatica.2019.108508

[5] B. Djeridane and J. Lygeros, "Neural approximation of pde solutions: An application to reachability computations," in *IEEE Conference on Decision and Control*, 2006, pp. 3034–3039.

[6] P.-A. Coquelin, S. Martin, and R. Munos, "A dynamic programming approach to viability problems," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 178–184.

[7] F. Jiang, G. Chou, M. Chen, and C. J. Tomlin, "Using neural networks to compute approximate and guaranteed feasible hamilton-jacobi-bellman pde solutions," 2016. [Online]. Available: https://www.arxiv.org/abs/1611.03158

[8] V. Rubies-Royo and C. Tomlin, "Recursive Regression with Neural Networks: Approximating the HJI PDE Solution," in *International Conference on Learning Representations*, 2017.

[9] K. C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and Liveness Guarantees through Reach-Avoid Reinforcement Learning," *Robotics: Science and Systems*, 2021.

[10] C. Dawson, S. Gao, and C. Fan, "Safe Control With Learned Certificates : A Survey of Neural Lyapunov , Barrier , and Contraction Methods for Robotics and Control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1749–1767, 2023.

[11] Y. Zhou, D. Li, Y. Xi, and Y. Xu, "Data-driven approximation for feasible regions in nonlinear model predictive control," 2020. [Online]. Available: https://arxiv.org/abs/2012.03428

[12] A. La Rocca, M. Saveriano, and A. Del Prete, "VBOC: Learning the Viability Boundary of a Robot Manipulator using Optimal Control," *IEEE Robotics and Automation Letters*, 2023.

[13] K. Nakamura, L. Peters, and A. Bajcsy, "Generalizing safety beyond collision-avoidance via latent-space reachability analysis," *arXiv preprint arXiv:2502.00935*, 2025.

[14] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[15] G. Lunardi, A. L. Rocca, M. Saveriano, and A. D. Prete, "Receding-Constraint Model Predictive Control using a Learned Approximate Control-Invariant Set," in *IEEE International Conference on Robotics and Automation*, 2024.

[16] E. Fontanari, G. Lunardi, M. Saveriano, and A. D. Prete, "Parallel-Constraint Model Predictive Control: Exploiting Parallel Computation for Improving Safety," in *IEEE International Conference on Robotics and Automation*, 2025.

[17] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[18] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "Acados: a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, pp. 147–183, 2019.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[20] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryll, "Learning for casadi: Data-driven models in numerical optimization," in *6th Annual Learning for Dynamics & Control Conference*. PMLR, 2024, pp. 541–553.

[21] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal stability of real-time iteration scheme for nonlinear model predictive control," *IEE Proceedings-Control Theory and Applications*, vol. 152, no. 3, pp. 296–308, 2005.

[22] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning Control Barrier Functions from Expert Demonstrations," in *Proceedings of the IEEE Conference on Decision and Control*, 2020, pp. 3717–3724.