2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Safe Imitation Learning of Nonlinear Model Predictive Control for Flexible Robots

Shamil Mamedov[1,2], Rudolf Reiter[3], Seyed Mahdi B. Azad[4], Ruan Viljoen[1,2], Joschka Boedecker[4,6], Moritz Diehl[3,5], Jan Swevers[1,2]

***Abstract*— Flexible robots may overcome some of the industry's major challenges, such as enabling intrinsically safe human-robot collaboration and achieving a higher payload-to-mass ratio. However, controlling flexible robots is complicated due to their complex dynamics, which include oscillatory behavior and a high-dimensional state space. Nonlinear model predictive control (NMPC) offers an effective means to control such robots, but its significant computational demand often limits its application in real-time scenarios. To enable fast control of flexible robots, we propose a framework for a safe approximation of NMPC using imitation learning and a predictive safety filter. Our framework significantly reduces computation time while incurring a slight loss in performance. Compared to NMPC, our framework shows more than an *eightfold* improvement in computation time when controlling a three-dimensional flexible robot arm in simulation, all while guaranteeing safety constraints. Notably, our approach outperforms state-of-the-art reinforcement learning methods. The development of fast and safe approximate NMPC holds the potential to accelerate the adoption of flexible robots in industry. The project code is available at: tinyurl.com/anmpc4fr**

I. Introduction

In recent years, flexible robots have drawn increasing attention as they may hold the key to solving significant problems in industry. These problems include improving the payload-to-mass ratio, as well as making robots intrinsically safer to facilitate human-robot collaboration [1]. The primary reason why flexible robots have not yet been widely adopted is their inherent flexibility, which causes oscillations and static deflections, complicating modeling and control.

Modeling flexible robots is challenging because they possess an infinite number of degrees of freedom (DOF) and are governed by nonlinear partial differential equations. Spatial discretization converts these equations into ordinary differential equations, rendering them more suitable for control and trajectory planning. Although various modeling methods exist for flexible robots [2, 3, 4], this paper adopts a lumped parameter approach [4, 5, 6, 7], which balances computational efficiency and accuracy. Having a model lays the groundwork for leveraging model-based methods, particularly optimal control, to ensure safety and high performance. An alternative approach is model-free control, particularly model-free reinforcement learning (RL). However, model-free approaches often suffer from sample inefficiency [8].

This paper specifically focuses on using model predictive control (MPC) for the output regulation problem of flexible robots: the task of reaching a desired end-effector position from an arbitrary initial configuration. The inherent nonlinearity of flexible robot dynamics necessitates the adoption of nonlinear MPC (NMPC). However, deploying NMPC for real-time control of flexible robots is often impossible due to slow computation time caused by the high-dimensional state space and nonlinear dynamics. Prior attempts to mitigate this issue have leveraged linearized MPC approaches, yielding overly conservative control strategies. In contrast, we propose an alternative approach that approximates NMPC using a neural network (or any suitable function approximator), a technique known as imitation learning. A major drawback of the approximated policy is the lack of safety guarantees. We address this by incorporating a safety filter [9] that modifies the candidate action proposed by the learned controller if it results in a constraint violation.

To summarize, this paper leverages ideas from optimal control, imitation learning, and safety-critical control, proposing a framework for the accurate and safe output regulation of flexible robots. This framework represents the main contribution of this paper and has been validated through extensive simulation experiments. Specifically, we demonstrate that our framework not only significantly reduces the computational time of control action but also effectively filters out unsafe control actions, closely approaching the performance of NMPC. Moreover, it outperforms state-of-the-art RL algorithms.

II. Related work

We organize the related work around key components of the proposed framework.

A. Modeling flexible robots

Spatial discretization serves as the primary tool for deriving computationally tractable models of flexible robots. There are three main methods for discretizing a flexible link: the assumed mode method (AMM) [2, 10], the finite element method (FEM) [3, 11] and the lumped parameter method (LPM) [4, 5, 6, 7]. All methods generally assume small deformations and employ the linear theory of elasticity. The FEM is the most accurate among the three but results in a higher number of differential equations. The AMM is often utilized for one DOF flexible robots; for robots with higher

[1]The MECO Research Team, KU Leuven, Leuven, Belgium.
[2]The DMMS Lab, Flanders Make, Leuven, Belgium.
[3]Department of Microsystems Engineering, University of Freiburg, Freiburg, Germany.
[4]Department of Computer Science, University of Freiburg, Freiburg, Germany.
[5]Department of Mathematics, University of Freiburg, Freiburg, Germany.
[6]BrainLinks-BrainTools, University of Freiburg, Freiburg, Germany.

DOF, the choice of boundary conditions becomes nontrivial [12]. The LPM is the simplest among all, but tuning the parameters of such models may require significant effort. In this paper, we leverage the LPM following a formulation defined in [13] because of its simplicity and ability to exploit existing efficient rigid-body dynamics algorithms.

B. Optimal control of flexible robots

Numerous control methods have been proposed for controlling flexible robots, including optimal control methods. Green et al. [10] applied a linear quadratic controller for trajectory tracking control of a two-link flexible robot, utilizing linearized dynamics. Both [14] and [15] used MPC to control a single-link flexible robot and four-link flexible mechanisms, respectively. In these papers, the authors linearized the dynamics and used linear MPC, highlighting the difficulty in developing fast NMPC for robots with high-dimensional dynamics. Contrary to the approaches taken by [14, 15], we first formulate an NMPC for the output regulation of flexible robots that fully leverages nonlinear dynamics and then approximate it.

C. Approximating model predictive controllers

To broaden the application of computationally demanding NMPC to fast robotics systems, recent efforts have aimed at approximating NMPC with neural networks (NN). Both [16] and [17] approximated robust and nominal NMPC for motion planning in fully-actuated rigid robot arms, respectively. Nurbayeva et al. [17] did not explicitly address the safety of the approximated policy, while Nubert et al. [16] applied a statistical validation technique to ensure the safety of the approximated NMPC, despite its time-consuming nature. Carius et al. [18] proposed a policy search method guided by NMPC without safety considerations.

Approximating NMPC with a NN falls within the broader domain of imitation learning (IL). A straightforward IL method known as behavioral cloning (BC) [19] approximates NMPC (the expert policy) by addressing a supervised regression problem on a dataset $\mathcal{D}$ containing state-action pairs. However, BC is limited as it can occasionally make mistakes, causing it to venture into areas of the state space not covered in $\mathcal{D}$, resulting in unpredictable behavior and poor performance. The DAgger algorithm [20] overcomes BC's shortcomings by collecting additional expert demonstrations $\mathcal{D}_{\text{new}}$ as the robot interacts with the environment under the state distribution induced by the learned policy. Essentially, DAgger iteratively learns from the expert to mitigate the learned policy's approximation errors. In contrast to BC and DAgger, Inverse Reinforcement Learning (IRL) aims to infer the expert's reward function from the dataset $\mathcal{D}$ and train a policy based on this learned reward function. Although IRL can yield more robust policies, these methods are often difficult to train due to their adversarial nature [21].

Given that the expert NMPC controller can be readily queried by the DAgger during training, we incorporate DAgger into the proposed framework, unlike [16, 17, 18] who employed BC. Our ablation studies (Section V-D) demonstrate DAgger's superiority over other imitation learning algorithms.

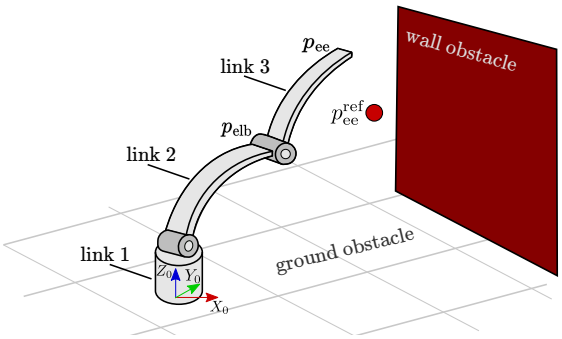

Fig. 1: An illustration of the problem addressed in this paper. The controller has to move the flexible robot's end-effector p_{ee} to the goal position $p_{\text{ee}}^{\text{ref}}$ while guaranteeing safety in terms of preventing the robot from colliding with the *wall* and *ground* obstacles.

D. Guaranteeing safety of learned controllers

While approximation reduces the computational demands of deploying NMPC, it does not inherently guarantee safety. Recently, several methods have been proposed to ensure the safety of learned black-box controllers [22]. Control barrier functions [23] and safety filters (SF) [9] stand out as two prominent approaches. The former requires an explicit definition of the safe set and operates reactively, while the latter implicitly defines the safe set and acts predictively. Given the challenge of explicitly specifying safe sets for flexible robots, this paper utilizes a SF: an NMPC scheme that evaluates a candidate action provided by the learned controller to verify that it can drive the system to a safe terminal set. If so, the action is directly applied to the system; otherwise, it is modified as little as possible to ensure safety.

III. Problem Statement

Consider the three DOF flexible robot arms shown in Fig. 1, inspired by the flexible robots TUDOR [24] and ELLA [6]. The first link is rigid, while the second and third links are flexible, sharing identical dimensions and material properties. The joints are assumed to be rigid and directly actuated, meaning that the actuators do not include gearboxes. Let $\boldsymbol{x}_{\text{tr}} \in \mathbb{R}^{n_{x_{\text{tr}}}}$, $\boldsymbol{u} \in \mathbb{R}^3$ and $\boldsymbol{z} := \boldsymbol{p}_{\text{ee}} \in \mathbb{R}^3$ denote the *true* state, control inputs, and controlled output (end-effector position) of a flexible robot, respectively. Furthermore, let the dynamics and output equations be defined in state-space form as follows:

$$\dot{\boldsymbol{x}}_{\text{tr}} = \boldsymbol{f}_{\text{tr}}(\boldsymbol{x}_{\text{tr}}, \boldsymbol{u}); \quad \boldsymbol{z} = \boldsymbol{h}_{\text{tr}}(\boldsymbol{x}_{\text{tr}}). \tag{1}$$

where $\boldsymbol{f}_{\text{tr}}(\boldsymbol{x}_{\text{tr}}, \boldsymbol{u})$ and $\boldsymbol{h}_{\text{tr}}(\boldsymbol{x}_{\text{tr}})$ represent *true* nonlinear dynamics and controlled output maps, respectively. The problem addressed in this paper is accurate and real-time feasible output regulation: steering the robot from an initial rest state $\boldsymbol{x}_{\text{tr},0}$ and corresponding output $\boldsymbol{z} = \boldsymbol{h}_{\text{tr}}(\boldsymbol{x}_{\text{tr},0})$ to a final output $\boldsymbol{z}^{\text{ref}} := \boldsymbol{p}_{\text{ee}}^{\text{ref}}$. The controller must adhere to the robot's constraints, such as velocity and input/torque limits while avoiding obstacles, including a *wall* and *ground*

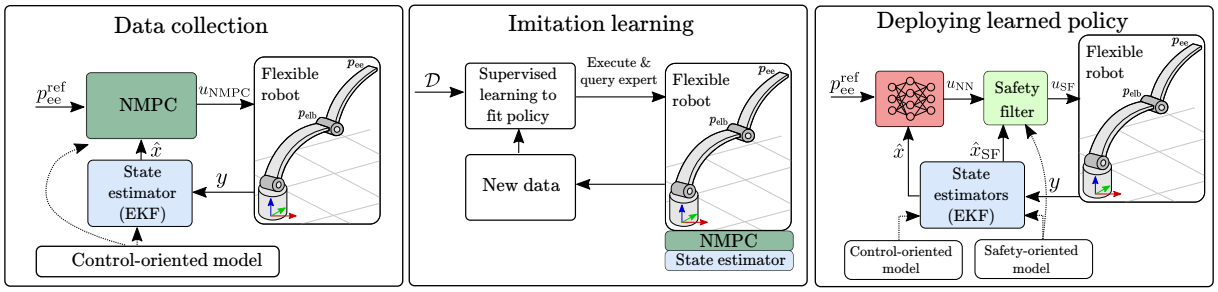

Fig. 2: Proposed framework for safely approximating NMPC policy by combining imitation learning of NMPC and a safety filter.

obstacles, as illustrated in Fig. 1. The measurements $\boldsymbol{y} = [\boldsymbol{q}_\mathrm{a}^\top \ \dot{\boldsymbol{q}}_\mathrm{a}^\top \ \boldsymbol{p}_\mathrm{ee}^\top]^\top \in \mathbb{R}^9$ include the positions $\boldsymbol{q}_a \in \mathbb{R}^3$ and velocities $\dot{\boldsymbol{q}}_a \in \mathbb{R}^3$ of the actuated joints, as well as the position of the end effector (EE) $\boldsymbol{p}_\mathrm{ee}$, to monitor elastic deformations.

IV. Method

NMPC can effectively address the constrained output regulation problem for flexible robots by formulating an optimal control problem (OCP) as a nonlinear program (NLP) and leveraging nonlinear optimization solvers to find a sequence of control inputs that steers the robot towards the desired output [25]. This NLP is then repeatedly solved online for a receding horizon. However, due to the complexity of a flexible robot's dynamics and its high-dimensional state space, solving the optimization problem becomes too slow for real-time control.

To mitigate this issue and make NMPC available for flexible robots, we propose a framework, as illustrated in Fig. 2, that safely approximates NMPC. The approach involves three steps. First, it formulates NMPC for the flexible robot, denoted by $\boldsymbol{\pi}_\mathrm{NMPC}$, and generates a dataset $\mathcal{D}$ by deploying $\boldsymbol{\pi}_\mathrm{NMPC}$ in simulation. Second, it employs the imitation learning algorithm to approximate $\boldsymbol{\pi}_\mathrm{NMPC}$ using $\mathcal{D}$, along with an additional dataset $\mathcal{D}_\mathrm{new}$ collected during interaction with the robot and the expert NMPC, thereby yielding $\boldsymbol{\pi}_\mathrm{NN}$. Third, it deploys $\boldsymbol{\pi}_\mathrm{NN}$ using a safety filter to ensure safety and prevent constraint violations.

The simulation, control-oriented, and safety-oriented models differ in state space dimensions because of the different spatial discretizations employed. Specifically, the simulation utilizes a higher fidelity model with finer spatial discretization. Consequently, an extended Kalman filter (EKF) is used to estimate the states of the control- and safety-oriented models based on measurements obtained from the simulation. The following subsections describe all the components of the framework.

A. Modeling and simulation of flexible robots

To model the robot, we employ an LPM, specifically the pseudo-rigid body method, which approximates a flexible link with a finite number of rigid links connected by elastic joints [7, 13], as illustrated in Fig. 3. This spatial discretization process involves two main steps. First, a flexible link is divided into n_seg segments, with their spring and damping properties lumped at a single point (primary division). Then,

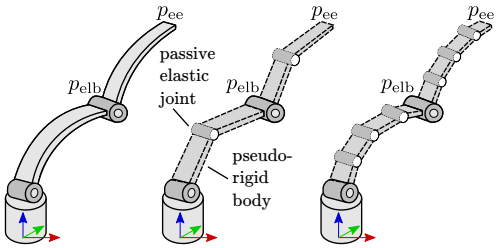

Fig. 3: Illustration of two different levels of spatial discretization: one-segment (middle) and three-segment (right) discretizations.

pseudo-rigid links are isolated between massless passive elastic joints (secondary division).

This spatial discretization enables the application of algorithms from rigid-body literature for approximating the kinematics and dynamics of the flexible robot [26]. We denote the vector of joint angles $\boldsymbol{q} = [\boldsymbol{q}_\mathrm{a}^\top \ \boldsymbol{q}_\mathrm{p}^\top]^\top$, with $\boldsymbol{q}_\mathrm{p} \in \mathbb{R}^{2n_\mathrm{seg}}$ representing passive elastic joint angles, typically modeled as spherical joints to capture the compliance in all directions. However, when compliance is primarily in one direction, as in our setup, simplifying the model to focus on this direction is beneficial. Our model specifically considers lateral bending, as shown in Fig. 3.

Applying the Lagrange method yields the final expression for the forward dynamics of the flexible manipulator

$$\ddot{\boldsymbol{q}} = \underbrace{\boldsymbol{M}(\boldsymbol{q})^{-1}\big\{\boldsymbol{B}\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{K}\boldsymbol{q} - \boldsymbol{D}\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\big\}}_{\mathrm{fwd}(\boldsymbol{q},\dot{\boldsymbol{q}})} \quad (2)$$

where $n_\mathrm{rb} = 3 + 2n_\mathrm{seg}$; $\boldsymbol{M} \in \mathbb{R}^{n_\mathrm{rb}\times n_\mathrm{rb}}$ is the symmetric inertia matrix; $\boldsymbol{K}, \boldsymbol{D} \in \mathbb{R}^{n_\mathrm{rb}\times n_\mathrm{rb}}$ are the constant diagonal stiffness and damping matrices, respectively; $\boldsymbol{C} \in \mathbb{R}^{n_\mathrm{rb}\times n_\mathrm{rb}}$ is the matrix of centrifugal and Coriolis forces, $\boldsymbol{g} \in \mathbb{R}^{n_\mathrm{rb}}$ is the vector of gravitational forces, $\boldsymbol{B} \in \mathbb{R}^{n_\mathrm{rb}\times 3}$ is the constant control jacobian and $\boldsymbol{\tau} \in \mathbb{R}^3$ is the torque vector. The model is converted to state-space form by defining $\boldsymbol{x} := \left[\boldsymbol{q}^\top \ \dot{\boldsymbol{q}}^\top\right]^\top$ and $\boldsymbol{u} := \boldsymbol{\tau}$

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u};n_\mathrm{seg}) = \left[\dot{\boldsymbol{q}}^\top \ \mathrm{fwd}(\boldsymbol{q},\dot{\boldsymbol{q}})^\top\right]^\top. \quad (3)$$

Similarly, the forward kinematics of a serial chain provides expressions for computing the positions of the end-effector $\boldsymbol{p}_\mathrm{ee}$ and the elbow of the robot:

$$\boldsymbol{p}_\mathrm{ee} = \boldsymbol{h}(\boldsymbol{x}), \quad \boldsymbol{p}_\mathrm{elb} = \boldsymbol{\kappa}(\boldsymbol{x}). \quad (4)$$

Note that the model's accuracy and computational complexity depend on the level of spatial discretization n_seg. Therefore, we consider the model with $n_\mathrm{seg} = 10$ as the ground truth dynamics (1) used for simulation (further

L [cm]	a [cm]	h [cm]	ρ [kg/m^3]	E [GPa]	$\eta \times 10^9$
50	5	0.2	7870	190	0.85

TABLE I: Material properties of the flexible links. L, a, h are length, width, and height, respectively. ρ is the density, E is the Young's modulus and η is the normal damping coefficient.

increasing n_{seg} does not improve simulation accuracy). For the control-oriented and safety-oriented models used by the NMPC and safety filter, respectively, we employed coarser but computationally faster models. Material properties of the flexible links are presented in Table I.

For the efficient implementation of (3), we use the articulated body algorithm for forward dynamics and the forward path of the recursive Newton-Euler algorithm for forward kinematics, both generated by Pinocchio [27] as CasADi [28] functions. To simulate the flexible robot, we employ an implicit integrator (the backward differentiation formula, as implemented in CVODES [29]), due to the stiffness of (3). Additionally, to simulate sensor noise, we add Gaussian noise to all measurements $\boldsymbol{y}$. Considering variations in n_{seg} between the control and the simulation models, and the presence of measurement noise, we implemented an EKF as state estimator to infer control-oriented and safety-oriented models states from the measurements $\boldsymbol{y}$.

B. NMPC for output regulation of flexible robots

In robotics, an OCP is initially formulated in continuous time, after which it is discretized and represented as an NLP [25], which is solved at each iteration of NMPC. Various techniques are available for translating OCP into an NLP, and this paper adopts the direct multiple shooting approach [30], where the decision variables are both states and control actions. A general NLP for output regulation of a flexible robot can be expressed as follows:

$$\underset{\boldsymbol{X},\boldsymbol{U},\boldsymbol{Z},\boldsymbol{\Sigma}}{\text{minimize}} \quad L(\boldsymbol{X},\boldsymbol{U},\boldsymbol{Z},\boldsymbol{\Sigma}) \tag{5a}$$

subject to

$$\boldsymbol{x}_0 = \hat{\boldsymbol{x}}, \qquad \boldsymbol{\Sigma} + \Delta \geq 0, \quad \boldsymbol{x}_N \in S^t, \tag{5b}$$

$$\boldsymbol{0} = \boldsymbol{\Phi}(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k, \boldsymbol{u}_k), \qquad k = 0,\dots,N-1, \tag{5c}$$

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}_k), \qquad k = 0,\dots,N, \tag{5d}$$

$$\underline{\boldsymbol{x}} - \sigma_{x,k} \leq \boldsymbol{x}_k \leq \overline{\boldsymbol{x}} + \sigma_{x,k}, \quad k = 0,\dots,N, \tag{5e}$$

$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}_k \leq \overline{\boldsymbol{u}}, \qquad k = 0,\dots,N-1, \tag{5f}$$

$$\boldsymbol{z} \in \mathcal{F}(\sigma_{\text{ee}}), \qquad k = 0,\dots,N, \tag{5g}$$

$$\boldsymbol{\kappa}(\boldsymbol{x}_k) \in \mathcal{F}(\sigma_{\text{elb}}), \qquad k = 0,\dots,N. \tag{5h}$$

In this formulation, N represents the prediction horizon, the decision variables $\boldsymbol{X} = [\boldsymbol{x}_0,\dots,\boldsymbol{x}_N] \in \mathbb{R}^{n_x \times (N+1)}$ and $\boldsymbol{U} = [\boldsymbol{u}_0,\dots,\boldsymbol{u}_{N-1}] \in \mathbb{R}^{n_u \times N}$ correspond to states and control inputs, respectively. Algebraic decision variables $\boldsymbol{Z} = [\boldsymbol{z}_0,\dots,\boldsymbol{z}_N] \in \mathbb{R}^{n_z \times N}$ represent the controlled output of the robot.

The constraints include: the discretized nonlinear dynamics of the robot (5c) as implicit integration condition; upper $\overline{\boldsymbol{u}}$ and lower $\underline{\boldsymbol{u}}$ bounds on control actions (torques) (5f); upper $\overline{\boldsymbol{x}}$ and lower $\underline{\boldsymbol{x}}$ bounds on states (5e), softened via slack variables $\sigma_{x,k} \in \mathbb{R}^+$; safety-critical, obstacle avoidance constraints for the EE and elbow positions, (5g) and (5h), respectively. The feasible set $\mathcal{F}(\sigma)$ in (5g) and (5h) is formulated as a convex set that results from the intersection of two half-spaces:

$$\mathcal{F}(\sigma) = \left\{ \boldsymbol{p} \in \mathbb{R}^3 \,\middle|\, \begin{matrix} \boldsymbol{w}_{\text{wall}}^\top (\boldsymbol{p} - \boldsymbol{b}_{\text{wall}}) \geq -\sigma \\ \boldsymbol{w}_{\text{ground}}^\top (\boldsymbol{p} - \boldsymbol{b}_{\text{ground}}) \geq -\sigma \end{matrix} \right\}, \tag{6}$$

where $\sigma \in \mathbb{R}^+$ is a slack parameter used for numerical robustness in optimization algorithms and which is zero in the optimal solution, $\boldsymbol{w}_i$ and $\boldsymbol{b}_i$ are the hyperplane parameters, with $i \in \{\text{wall}, \text{ground}\}$. All the slack decision variables are represented by $\boldsymbol{\Sigma} = [\boldsymbol{\sigma}_0,\dots,\boldsymbol{\sigma}_N]^\top \in \mathbb{R}^{3\times N}$, with $\boldsymbol{\sigma}_k = [\sigma_{x,k}\ \sigma_{\text{ee},k}\ \sigma_{\text{elb},k}]^\top \in \mathbb{R}^3$.

To mitigate constraint violations due to noisy measurements and model mismatch, we performed a simple heuristic-based constraint tightening via slack variables. Specifically, we established safety margins $\boldsymbol{\delta}_x \in \mathbb{R}^{n_x}$ for the state and safety-critical constraints $\boldsymbol{\delta}_{\text{ee}}$ and $\boldsymbol{\delta}_{\text{elb}}$. The tightening of the state, output, and elbow constraints was formulated by setting bounds on the corresponding slack variables σ_x, σ_{ee} and σ_{elb}, respectively. To concisely formulate the slack bounds (5b), we utilized the matrix $\Delta = [\delta_x\ \delta_{\text{ee}}\ \delta_{\text{elb}}] \otimes \boldsymbol{1}_{1\times N}$, which repeats the vector $[\delta_x\ \delta_{\text{ee}}\ \delta_{\text{elb}}]$ across N times.

The objective function L penalizes the deviation from the reference controlled output position $\boldsymbol{z}^{\text{ref}}$, and, as a form of regularization, it also penalizes deviations from the reference state $\boldsymbol{x}^{\text{ref}}$ and reference torque $\boldsymbol{u}^{\text{ref}}$ using an ℓ_2 norm:

$$\begin{aligned} L(\boldsymbol{X},\boldsymbol{U},\boldsymbol{Z},\boldsymbol{\Sigma}) = \sum_{k=0}^{N-1} \left\|\boldsymbol{x}_k - \boldsymbol{x}^{\text{ref}}\right\|_{\boldsymbol{Q}}^2 + \left\|\boldsymbol{u}_k - \boldsymbol{u}^{\text{ref}}\right\|_{\boldsymbol{R}}^2 + \\ \left\|\boldsymbol{z}_k - \boldsymbol{z}^{\text{ref}}\right\|_{\boldsymbol{P}}^2 + \|\boldsymbol{\sigma}_k\|_{\boldsymbol{S}}^2 + \|\boldsymbol{\sigma}_k\|_{1,s} + \left\|\boldsymbol{x}_N - \boldsymbol{x}_N^{\text{ref}}\right\|_{\boldsymbol{Q}_N}^2 + \\ \left\|\boldsymbol{z}_N - \boldsymbol{z}^{\text{ref}}\right\|_{\boldsymbol{P}_N}^2 + \|\boldsymbol{\sigma}_N\|_{\boldsymbol{S}}^2 + \|\boldsymbol{\sigma}_N\|_{1,s}, \end{aligned} \tag{7}$$

with $\boldsymbol{Q} = \text{diag}(\boldsymbol{q}) \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{R} = \text{diag}(\boldsymbol{r}) \in \mathbb{R}^{n_u \times n_u}$ and $\boldsymbol{P} = \text{diag}(\boldsymbol{p}) \in \mathbb{R}^{n_z \times n_z}$ being stage cost weighting matrices; and $\boldsymbol{Q}_N \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{P}_N \in \mathbb{R}^{n_z \times n_z}$ being terminal cost weighting matrices. Additionally, the slack variables $\boldsymbol{\Sigma}$ are penalized in the cost function by ℓ_1- and squared ℓ_2-norms by weights $\boldsymbol{s} \in \mathbb{R}^3$ and $\boldsymbol{S} \in \mathbb{R}^{3\times 3}$, respectively. The latter is used to facilitate convergence.

Since the optimization problem was formulated for a finite horizon, a simple safe terminal set $S^t = \{\boldsymbol{x} \in \mathbb{R}^{n_x} | \boldsymbol{H}\boldsymbol{x} = 0\}$ was used to guarantee recursive feasibility. S^t corresponds to zero joint velocities and the matrix $\boldsymbol{H}$ selects the angular velocity states $\dot{\boldsymbol{q}} = \boldsymbol{H}\boldsymbol{x}$ of the state vector $\boldsymbol{x}$.

For solving (5), we used the real-time NMPC solver `acados` [31] with the `HPIPM` QP solver [32]. Hyperplane parameters for the wall are defined by $\boldsymbol{w}_{\text{wall}} = [0\ 1\ 0]^\top$ and $\boldsymbol{b}_{\text{wall}} = [0\ -0.15\ 0.5]^\top$ and for the ground by $\boldsymbol{w}_{\text{ground}} = [0\ 0\ 1]^\top$ and $\boldsymbol{b}_{\text{ground}} = [0\ 0\ 0]^\top$, respectively. The remaining parameters of the NMPC are provided in the project's repository.

C. Imitation learning of the NMPC

The aim of IL within the proposed framework is to approximate the solution of (5), also referred as the expert

policy, by a NN using a dataset of expert demonstrations $\mathcal{D} = \{(\boldsymbol{u}_{\text{NMPC},i}, \hat{\boldsymbol{x}}_{\boldsymbol{i}}, \boldsymbol{s}_i)\}_{i=1}^{N}$, where $\boldsymbol{u}_{\text{NMPC},i}$ is the control action of NMPC, $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ is a state estimate and $\boldsymbol{s}_i$ represents a context. The context provides additional information about the system that is not included in state-action pairs. It includes the current and desired positions of the end-effector $\boldsymbol{p}_{\text{ee}}$ and $\boldsymbol{p}_{\text{ee}}^{\text{ref}}$, respectively, as well as the hyperplane representation of the safe-critical constraints $\boldsymbol{w}_i$ and $\boldsymbol{b}_i$ with $i \in \{\text{wall}, \text{ground}\}$. In addition to $\mathcal{D}$, DAgger collects a dataset $\mathcal{D}_{\text{new}}$ during interaction with the flexible robot and the expert NMPC.

We leveraged the `imitation` [33] implementation of DAgger, which started from a dataset $\mathcal{D}$ of 100 expert demonstrations and collected 500 more demonstrations during training. Key hyperparameters employed during training are provided in the project repository.

D. Safety filter

The safety filter can be interpreted as an implicit formulation of the safe set, which is closely related to the formulation of an NMPC. However, since the safety filter is used only for projecting the NN control actions into a safe set and not for performance, the model it uses is simpler and faster to integrate, and the prediction horizon can be shorter. The proposed safety filter policy $\boldsymbol{\pi}_{\text{SF}} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ projects the NN output $\boldsymbol{u}_{\text{NN}} \in \mathbb{R}^{n_u}$ to a safe control $\boldsymbol{u}_{\text{SF}} = \boldsymbol{\pi}_{\text{SF}}(\boldsymbol{x}_{\text{SF}}, \boldsymbol{u}_{\text{NN}}) \in \mathcal{U}_{\text{SF}} \subseteq \mathbb{R}^{n_u}$ by using the same formulation as in (5), but with modified cost L_{SF}, a lower fidelity model $\boldsymbol{x}_{\text{SF},i+1} = \boldsymbol{\Phi}_{\text{SF}}(\boldsymbol{x}_{\text{SF},i}, \boldsymbol{u}_{\text{SF},i})$ with just one segment ($n_{\text{seg}} = 1$), as shown in Fig. 3, and a shorter horizon. The safety filter cost function

$$L_{\text{SF}}(\boldsymbol{X}_{\text{SF}}, \boldsymbol{U}_{\text{SF}}, \boldsymbol{\Sigma}_{\text{SF}}) = \|\boldsymbol{u}_0 - \boldsymbol{u}_{\text{NN}}\|_{\boldsymbol{R}}^2 + L_{\text{REG}} \tag{8}$$

penalizes the deviation from the proposed control $\boldsymbol{u}_{\text{NN}}$, besides a regularization term

$$\begin{aligned} L_{\text{REG}}(\boldsymbol{X}_{\text{SF}}, \boldsymbol{U}_{\text{SF}}, \boldsymbol{\Sigma}_{\text{SF}}) = &\sum_{k=1}^{N_{\text{SF}}-1} \|\boldsymbol{u}_k\|_{\boldsymbol{R}_{\text{SF}}}^2 + \\ &\sum_{k=0}^{N_{\text{SF}}} \left\|\boldsymbol{x}_k - \boldsymbol{x}^{\text{ref}}\right\|_{\boldsymbol{Q}_{\text{SF}}}^2 + \|\boldsymbol{\sigma}_k\|_{\boldsymbol{S}_{\text{SF}}}^2 + \|\boldsymbol{\sigma}_k\|_{1,s_{\text{SF}}}, \end{aligned} \tag{9}$$

with small weights $\boldsymbol{Q}_{\text{SF}}$, $\boldsymbol{R}_{\text{SF}}$, $\boldsymbol{S}_{\text{SF}}$, and used for numerical robustness. While (8) looks similar to (5), due to a simpler model with fewer segments, it can be solved much faster.

V. Experimental validation

To evaluate the proposed approach, we conducted extensive simulation experiments where the initial configuration of the robot and desired goal position of the EE were randomly sampled. The goal positions were specifically chosen near obstacles to test safety. Additionally, we evaluated the policies' robustness against model uncertainties – reduction in the stiffness parameters of the flexible robot by 10%, in particular the $\boldsymbol{K}$ matrix in (2). For evaluation, we considered metrics such as the final distance to the goal at the end of each episode/rollout, policy evaluation time (inference time), and the number of constraint violations during an episode.

n_{seg}	$d_{\mathcal{G}_\epsilon}/d^*_{\mathcal{G}_\epsilon}$	$t_{\mathcal{G}_\epsilon}/t^*_{\mathcal{G}_\epsilon}$	$\bar{t}_{\text{NMPC}}/\bar{t}^*_{\text{NMPC}}$	$t^{\max}_{\text{NMPC}}/t^{\max,\,*}_{\text{NMPC}}$
3	1.008	0.955	1.653	1.589
5	1.026	0.974	2.95	2.940

TABLE II: Performance comparison of NMPC controllers with different models given by the number of segments n_{seg}. $\bar{t}_{\text{NMPC}}$ and $t^{\max}_{\text{NMPC}}$ are average and maximum NMPC computation times, respectively. $t_{\mathcal{G}_\epsilon}$ is the time to reach an ϵ-ball around the reference output; $d_{\mathcal{G}_\epsilon}$ is the radius of a ball that includes all points after 3 seconds. Evaluation metrics are measured relative to the NMPC with $n_{\text{seg}} = 2$, denoted with superscript *.

A. Reinforcement learning baseline

As model-free RL baselines, we trained SAC [34] and PPO [35] as the state-of-the-art representatives of offline and online RL algorithms, respectively. Controlling the flexible robot by means of RL is not the main focus of this paper; thus, we did not engineer the reward function r. Moreover, reward engineering is difficult in practice and can lead to unintended consequences [36]. Instead, we simply used the Euclidean distance between the robot's EE and the target EE positions, $r = -\|\boldsymbol{p}_{\text{ee}} - \boldsymbol{p}_{\text{ee}}^{\text{ref}}\|_2$. We penalized the agent with r_{pen} if it violated the safety-critical constraints. Both algorithms were trained for 2M gradient steps. When tested on 100 regulation tasks, SAC considerably outperformed PPO. Therefore, in further experiments, only SAC was used as an RL baseline.

B. Selecting a control-oriented model for NMPC

To analyze the influence of the model fidelity (*i.e.*, the number of segments) on the controller performance, we compared the NMPC for $n_{\text{seg}} \in \{0, 1, 2, 3, 5, 10\}$. The performance was evaluated by measuring the computation time t_{NMPC}, the time $t_{\mathcal{G}_\epsilon}$ that NMPC needed to steer the EE into an epsilon ball $\left\|\boldsymbol{z}(t) - \boldsymbol{z}^{\text{ref}}\right\| \leq \epsilon$ around the reference position $\boldsymbol{z}^{\text{ref}}$ ($\epsilon = 10^{-3}$ in our case) and the smallest ball with radius d that includes all points after 3 seconds

$$d_{\mathcal{G}_\epsilon} = \min_d \left\|\boldsymbol{z}(t) - \boldsymbol{z}^{\text{ref}}\right\| \leq d \qquad \text{s.t. } t \geq 3\text{s}. \tag{10}$$

For $n_{\text{seg}} \in \{0, 1\}$ segments, the controller was unstable, and for ten segments, the computation time was unacceptably long. As shown in Tab. II, the average mean computation time $\bar{t}_{\text{NMPC}}$ and the maximum computation time $t^{\max}_{\text{NMPC}}$ significantly increase with the number of segments. However, the performance measured in terms of $t_{\mathcal{G}_\epsilon}$ and $d_{\mathcal{G}_\epsilon}$ does not significantly improve. Therefore, we selected $n_{\text{seg}} = 3$ as a compromise between performance and computation time for further use with IL algorithms.

C. Simulation results

1) Closed loop performance: Fig. 4 shows that NMPC outperforms the model-free RL algorithm SAC in the output regulation of the flexible robot both in terms of accuracy measured by final distance to goal and the number of constraint violations. While DAgger (*i.e.*, IL of NMPC) does not match the performance of the expert NMPC, it outperforms SAC.

Both DAgger and SAC algorithms violate safety-critical constraints, such as the wall and the ground, as shown

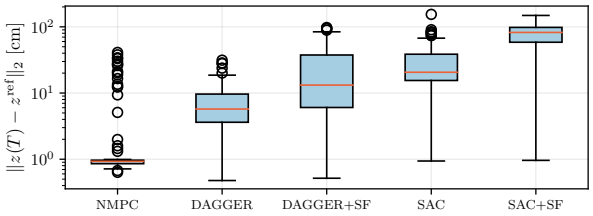

Fig. 4: The distribution of final distance-to-goal ($\|z(T) - z^{\text{ref}}\|_2$ with T being the episode duration) of the considered algorithms on 100 test tasks with and without safety filter.

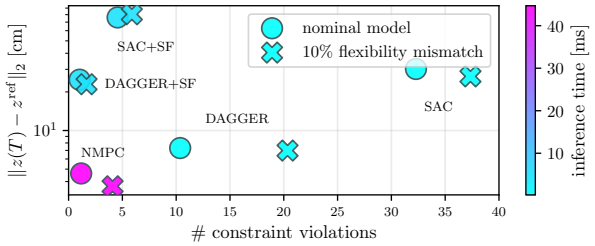

Fig. 5: Performance of the considered controllers with and without model-plant mismatch. This mismatch is implemented by reducing Young's modulus of the simulation model by 10%. The marker colors are defined by the inference time (policy evaluation time).

in Fig. 5. Importantly, both algorithms operate without explicit awareness of these constraints. RL agents learn about constraints indirectly through a scalar reward function, necessitating reward engineering to balance objectives like safety and goal achievement. In contrast, DAgger implicitly acquires knowledge of constraints from the trajectories of the expert. Through imitation of safe π_{NMPC}, DAgger violates constraints less often than SAC.

Incorporating a SF significantly reduces constraint violations, albeit at the cost of decreased performance and increased computation time. With SAC, the SF does not entirely eliminate all constraint violations, suggesting that when combined with SAC, the SF should adopt an even more conservative approach. In the case of DAgger, introducing the SF substantially enhances safety, making it more comparable to NMPC. Importantly, our proposed framework achieves an *eightfold* reduction in evaluation/inference time compared to NMPC, making the proposed approach real-time feasible, as shown in Fig. 5.

2) Robustness: Notably, none of the policies were explicitly trained for robustness, except for model mismatch due to different spatial discretizations. All policies used the nominal model for generating demonstrations $\mathcal{D}$ and for interaction. Fig. 5 shows that changes in stiffness parameters of the simulation model affect both final distance-to-goal and constraint violation frequency. Despite the model-plant mismatch, the expert NMPC consistently maintains high performance. In contrast, SAC and DAgger exhibit a decline in performance, particularly regarding constraint violations. The performance of the proposed framework and SAC combined with SF remains relatively unchanged.

D. Ablation of IL algorithms

To approximate NMPC, we compared BC [19], DAgger [20] and three IRL methods: GAIL [37] and AIRL [38], and

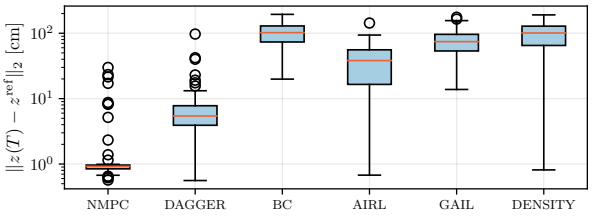

Fig. 6: Comparison of different IL algorithm for approximating NMPC in terms of distance-to-goal at the end of the episode.

a two-step IRL method denoted as Density. GAIL and AIRL both employ adversarial approaches to jointly learn the reward function and the policy. The two-step IRL method first learns the reward function using kernel density estimation on the expert demonstrations, similar to the approach in [39], then uses the learned reward function to train a SAC agent.

For training, we collected a dataset $\mathcal{D}$ of 100 expert demonstrations using NMPC and trained all algorithms for 2 million steps. To compare IL algorithms, we randomly generated 100 regulation tasks by sampling initial robot configurations and final end-effector positions within the robot's workspace. Figure 6 shows that DAgger significantly outperforms other algorithms, making it the preferred choice for IL of controllers like NMPC in simulation.

VI. CONCLUSIONS

This paper presents a novel framework for safely approximating NMPC for the output regulation of flexible robots. This framework integrates imitation learning with a safety filter to yield a controller that is not only computationally efficient but also accurate and safe. Initially, the framework employs DAgger for imitation learning, which approximates NMPC using a neural network, thereby significantly reducing the computation time as compared to NMPC. Then, a safety filter, formulated as a fast and simple NMPC, is employed to ensure obstacle avoidance and constraint satisfaction.

This framework was validated through extensive simulations involving a three-degree-of-freedom flexible manipulator, demonstrating a significant *eightfold* improvement in control action computation time. While this improvement entails some performance trade-offs, the proposed approach consistently outperformed the state-of-the-art reinforcement learning algorithm, SAC. Furthermore, we anticipate that the controller's performance will improve when more data is used to train the imitation learning component.

Additional insights gained from this research include: (i) the proposed framework demonstrates robustness to model-plant mismatches, and (ii) DAgger substantially outperforms other state-of-the-art imitation learning algorithms in approximating NMPC. Future research could extend this approach to trajectory tracking and control challenges in soft robotics.

ACKNOWLEDGMENT

This work was supported by several funding agencies: FWO-Vlaanderen through SBO project ELYSA for cobot applications (S001821N); DFG via Research Unit FOR 2401 and project 424107692 and by the EU via ELO-X 953348; the Carl Zeiss Foundation through the ReScaLe project.

REFERENCES

[1] A. De Luca and W. Book, *Robots with Flexible Elements*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 287–319.

[2] W. J. Book, "Recursive lagrangian dynamics of flexible manipulator arms," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 87–101, 1984.

[3] A. A. Shabana, *Dynamics of multibody systems*. Cambridge university press, 2020.

[4] T. Yoshikawa and K. Hosoda, "Modeling of flexible manipulators using virtual rigid links passive joints," *International Journal of Robotics Research*, vol. 15, no. 3, pp. 290–299, 1996.

[5] R. Franke, J. Malzahn, T. Nierobisch, F. Hoffmann, and T. Bertram, "Vibration control of a multi-link flexible robot arm with fiber-bragg-grating sensors," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3365–3370.

[6] P. Staufer and H. Gattringer, "State estimation on flexible robots using accelerometers and angular rate sensors," *Mechatronics*, vol. 22, no. 8, pp. 1043–1049, 2012.

[7] S. Moberg, E. Wernholt, S. Hanssen, and T. Brogårdh, "Modeling and parameter estimation of robot manipulators using extended flexible joint models," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, no. 3, p. 031005, 2014.

[8] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving Sample Efficiency in Model-Free Reinforcement Learning from Images," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10 674–10 681, May 2021.

[9] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, no. May, 2021.

[10] A. Green and J. Z. Sasiadek, "Dynamics and trajectory tracking control of a two-link robot manipulator," *Journal of Vibration and Control*, vol. 10, no. 10, pp. 1415–1440, 2004.

[11] W. Sunada and S. Dubowsky, "The application of finite element methods to the dynamic analysis of flexible spatial and co-planar linkage systems," 1981.

[12] A. Heckmann, "On the choice of boundary conditions for mode shapes in flexible multibody systems," *Multibody System Dynamics*, vol. 23, no. 2, pp. 141–163, 2010.

[13] E. Wittbrodt, I. Adamiec-Wójcik, and S. Wojciech, *Dynamics of flexible multibody systems: rigid finite element method*. Springer Science & Business Media, 2007.

[14] B. P. Silva, B. A. Santana, T. L. Santos, and M. A. Martins, "An implementable stabilizing model predictive controller applied to a rotary flexible link: An experimental case study," *Control Engineering Practice*, vol. 99, p. 104396, 2020.

[15] P. Boscariol, A. Gasparetto, and V. Zanotto, "Model predictive control of a flexible links mechanism," *Journal of Intelligent and Robotic Systems*, vol. 58, no. 2, pp. 125–147, 2010.

[16] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[17] A. Nurbayeva, A. Shintemirov, and M. Rubagotti, "Deep imitation learning of nonlinear model predictive control laws for a safe physical human–robot interaction," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 7, pp. 8384–8395, 2023.

[18] J. Carius, F. Farshidian, and M. Hutter, "MPC-net: A first principles guided policy search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, apr 2020.

[19] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proceedings of (NeurIPS) Neural Information Processing Systems*, D. Touretzky, Ed. Morgan Kaufmann, December 1989, pp. 305 – 313.

[20] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[21] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing training of generative adversarial networks through regularization," in *Neural Information Processing Systems*, 2017.

[22] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[23] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[24] J. Malzahn, M. Ruderman, A. Phung, F. Hoffmann, and T. Bertram, "Input shaping and strain gauge feedback vibration control of an elastic robotic arm," in *FConference on Control and Fault-Tolerant Systems (SysTol)*. IEEE, 2010, pp. 672–677.

[25] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 01 2017.

[26] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2001.

[27] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[28] 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.

[29] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 363–396, 2005.

[30] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[31] 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*, Oct 2021.

[32] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020, 21st IFAC World Congress.

[33] A. Gleave, M. Taufeeque, J. Rocamonde, E. Jenner, S. H. Wang, S. Toyer, M. Ernestus, N. Belrose, S. Emmons, and S. Russell, "imitation: Clean imitation learning implementations," arXiv:2211.11972v1 [cs.LG], 2022.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] J. Clark and D. Amodei, "Faulty reward functions in the wild," *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

[37] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[38] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," 2018.

[39] S. Choi, K. Lee, A. Park, and S. Oh, "Density matching reward learning," 2016.