

A Unified and Modular Model Predictive Control Framework for Soft Continuum Manipulators under Internal and External Constraints

Filippo A. Spinelli¹, and Robert K. Katzschmann¹

Abstract—Fluidically actuated soft robots have promising capabilities such as inherent compliance and user safety. The control of soft robots needs to properly handle nonlinear actuation dynamics, motion constraints, workspace limitations, and variable shape stiffness, so having a unique algorithm for all these issues would be extremely beneficial. In this work, we adapt *Model Predictive Control (MPC)*, popular for rigid robots, to a soft robotic arm called SoPrA. We address the challenges that current control methods are facing, by proposing a framework that handles these in a modular manner. While previous work focused on Joint-Space formulations, we show through simulation and experimental results that *Task-Space MPC* can be successfully implemented for dynamic soft robotic control. We provide a way to couple the *Piece-wise Constant Curvature* and *Augmented Rigid Body Model* assumptions with internal and external constraints and actuation dynamics, delivering an algorithm that unites these aspects and optimizes over them. We believe that a *MPC* implementation based on our approach could be the way to address most of model-based soft robotics control issues within a unified and modular framework, while allowing to include improvements that usually belong to other control domains such as machine learning techniques.

I. INTRODUCTION

In this work we have adapted a powerful model-based control method, *Model Predictive Control (MPC)*, to a Soft continuum Proprioceptive Arm (SoPrA) [1] that can perform dynamic motions and is easily scalable. Our model is based on the Piece-Wise Constant Curvature assumption and our control on the *Augmented Rigid Body Model* [2], [3].

Various model-based control methods have already been applied to soft robotic arms, such as the cascaded curvature controller based on Inverse Kinematics [4], the Curvature Dynamic Control [5] and the Sliding Mode Control (SMC) [6]. All those methods have shown promising results, but they suffer from a few issues. One is that a soft robotic arm is strongly limited by mechanical constraints both in actuation and in end-effector space; current controllers cannot deal with those limits, but *MPC* is inherently suited for constraint optimization. Della Santina et al. [7] have already made steps towards addressing this problem, developing a new algorithm to integrate internal constraints into the PCC control. With our work, we have provided an environment that would make use of the information extracted from an adaptation of such method to control soft arms with more control authority and awareness.

Another problem of most of pneumatically actuated soft-continuous arms is the need for limiting the number of

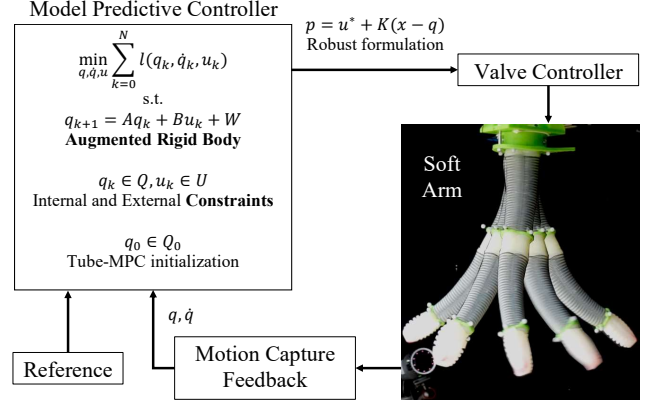


Fig. 1: Our proposed controller for fluidically actuated soft arms: a robust *Tube-based MPC* [8] with dynamics based on the *Augmented Rigid Body Model* [2], internal constraints due to actuation limits and external constraints due to task-space limitations.

PCC sections to avoid falling into an underactuated domain. However, increasing them would lead to better performances since the kinematic and dynamic approximation would be more accurate. Also, being able to deal with underactuated systems would allow to scale up current designs and map them to models with more PCC sections. As shown for example in [9], [10], *Model Predictive Control* is usually the choice adopted for such control problems.

MPC is an advanced model-based control technique. It is a form of optimal control with integrated feedback that directly takes into account system constraints, well suited for many model-based control problems since provides additional capabilities, including future trajectory prediction. The receding horizon approach allows to solve optimally feedback problems in a forward way, but the requirement of a finite horizon optimization leads to stability and feasibility problems, especially when dealing with disturbances. These issues have been addressed in different ways, for example with the *Tube-MPC* [8] or the *Indirect Feedback* [11].

MPC has been already applied successfully to soft-robotics platforms, for example in [12], [13]. They have modeled pressure dynamics and included it directly in the optimization problem, delivering an algorithm that exploits the additional information of an augmented state-space. Best et al. have also provided comparisons between *MPC* and other model-based controls [14], underlying the advantages of each methods: *MPC* can be as accurate as other controllers, while providing

¹F. A. Spinelli and R. K. Katzschmann are with the Soft Robotics Lab, ETH Zurich, Tannenstrasse 3, 8092 Zürich, Switzerland fspinelli@ethz.ch, rkk@ethz.ch

additional future knowledge and managing multiple DoFs at the same time. However, in all previous works analyzed they limited their efforts to joint-space tracking, choice that reduces possible practical applications. In this work, we moved further controlling the robot in task-space.

Starting from the previous research results, we have implemented a robust *MPC* able to deal with model uncertainties and used it for trajectory following tasks on SoPrA [1]. We have optimized over internal and external constraints, adopting few corrections to use standard *MPC* algorithms with the *Augmented Rigid Body* formulation.

This work aims to improve the *Augmented Rigid Body Model* technique [2], that takes into account geometric and inertial parameters while neglecting the pneumatic actuator dynamics and its inherent limits. By considering an augmented state-space vector as in [12] and adapting an offline algorithm similar to [7], we can enrich our control with the knowledge needed to better tackle motion tracking problems, as well as pick-and-place tasks, and generally extend the control authority to more complex domains. *MPC* has been used as unified framework to efficiently couple these techniques: kinematics and dynamics are derived from the *Augmented Rigid Body Model*, pressure dynamics accounted for with an augmented state vector, task-space limitations rewritten as polytopic constraints over curvature variables.

This work contributes:

- The *Soft-Robust MPC controller* based on the *Augmented Rigid Body Model* operating in task-space. The controller is fed with additional knowledge about actuation and task-space limitations to better fit into the soft-robotics control domain.
- A way to deal with internal constraints (actuation) and external constraints (obstacles) in a unified *MPC* environment.
- First real-world physical validations of *task-space MPC* on soft robotic arms.

II. MODEL

A. Mathematical Model

In [1] the kinematic and dynamic models for SoPrA have been formulated, via PCC and *Augmented Rigid Body* [2]. The dynamic equation for the soft arm is defined as:

$$Ap + J^T f = B(q)\ddot{q} + c(q, \dot{q}) + g(q) + Kq + D\dot{q} \quad (1)$$

where A is the allocation matrix, J^T the Jacobian from task-space to joint-space, B , c , g the inertial, Coriolis and gravitational terms, K , D additional stiffness and damping elements to account for the softness of the robot. The *Augmented Rigid Body* assumption consists in an inverse mapping from a mechanical linkage made of revolute and prismatic joints in space ξ to the curvature one, as in eq. (2). Please refer to [2], [5] and fig. 2 for more details.

$$\begin{aligned} B(q) &= J_m^T(q)B_\xi(m(q))J_m(q) \\ c(q, \dot{q}) &= J_m^T(q)c_\xi(m(q), J_m(q)\dot{q}) \\ g(q) &= J_m^T(q)g_\xi(m(q)) \end{aligned} \quad (2)$$

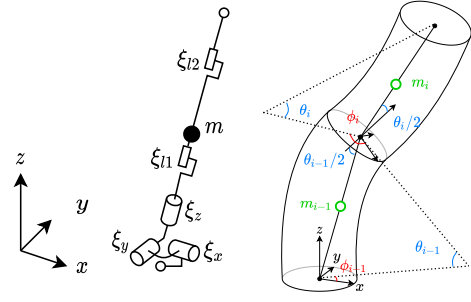


Fig. 2: *Augmented Rigid Body Model*: on the left the equivalent rigid joints and links, on the right the curvature-based space description. Adapted from [1].

Here $m(q)$ maps configuration between the two domains, and J_m is the Jacobian of the mapping:

$$J_m(q) = \frac{\partial m(q)}{\partial q} \quad (3)$$

However, differently from [2] only three symmetrically spaced chambers are present and the augmented state is reduced (see fig. 2, only 5 joints instead of 10 have been used to approximate the soft arm in 3D space). This choice leads to some adaptations in the control algorithm.

According to [1] the robotic arm is modeled with 2 curvature parameters per section, not the bending angle θ and off-plane rotation ϕ but rather two derived components:

$$\theta_x = \theta \cos(\phi) \quad \theta_y = \theta \sin(\phi) \quad (4)$$

This approach leads to a set of variables that doesn't reach singularity in the rest straight condition. According to this definition, the curvature vector we optimize over is:

$$q = [\theta_{x,1} \quad \theta_{y,1} \quad \theta_{x,2} \quad \dots \quad \theta_{y,N_{seg}N_{pcc}}]^T \quad \text{of size } q_{size} \quad (5)$$

Our arm has three pressures per segment, one per chamber, that can be controlled independently with positive values only. In order to avoid excessive constraints on actuation and the need of modifying the standard dynamic model, during control just two orthogonal and bi-directional pseudo-pressures are considered. Those are then used to geometrically reconstruct the actual input values.

B. MPC Model

Given eq. (1), in order to apply *MPC* we need to get a state-space equation for system evolution. It can be achieved with the following transformation:

$$\begin{aligned} \text{state} &= \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \quad \begin{bmatrix} \dot{q} \end{bmatrix} = \mathbf{A} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \mathbf{B}p + \mathbf{W} \\ \mathbf{A} &= \begin{bmatrix} 0_{q_{size} \times q_{size}} & I_{q_{size} \times q_{size}} \\ -B^{-1}K & -B^{-1}D \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} 0_{q_{size} \times 2segment s_{num}} \\ B^{-1}A \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 0_{q_{size} \times 1} \\ -B^{-1}(c + g) \end{bmatrix} \end{aligned} \quad (6)$$

We can write a state-space linearized system, that has to be updated each time the curvature configuration changes (so each control loop), since our formulation based on

Drake [15] would output only the numerical result of configuration-dependent matrices. One source of model errors is the Coriolis term, that is not in the common form $C(q, \dot{q})\dot{q}$ (that could be obtained for instance via Christoffel's symbols) but in a vectorial representation that can't be used to compute the A matrix. With this strong approximation we are assuming that curvature velocity affecting the Coriolis component changes slowly along the prediction horizon, very unlikely. However, since the order of magnitude of c term is usually much smaller than other quantities involved, the additional errors aren't extremely deteriorating for small speeds. The most critical issue in eq. (6) is that along the MPC prediction horizon the matrices won't update, so we can't consider a too long one. The inherent assumption is that state-dependent matrices would change slowly, as assumed for example also in [13].

Since *MPC* cannot work on continuous variables due to the need to execute the optimization processes, we converted our model into discrete-time. However, the discretization would lead to additional discrepancies between the real state evolution and the predicted one. Regardless the actual sampling time chosen T_s , the transformations needed are:

$$A_d = e^{A \times T_s} \quad B_d = A^{-1}(A_d - I_{2q_{size} \times 2q_{size}})B \quad W_d = W \times T_s \quad (7)$$

The whole process is computed once per control cycle. Selecting a small sampling time would result in an accurate approximation, but could produce real-time issues if the *MPC* isn't able to perform all the required computations within that frame; a large sampling time would on contrary introduce large model errors and reduce the controller's noise rejection ability.

In order to design a controller able to operate in task-space, we need the forward kinematics to map the curvature variables on which the model is based to the cartesian space where motion tracking tasks are defined. Since our state-space is made by θ_x and θ_y , we can just use a chain of simple rotations. The transformation we used for a two segments arm with one PCC section each is the following:

$$E_q = R_1(l_{s1}) + R_2(l_{c1}) + R_3(l_{s2}) + R_4(l_{c2}) \quad (8)$$

with $R_i = R_{i-1} \text{Rot}_y(-\frac{\theta_{x,[i/2]}}{2}) \text{Rot}_x(-\frac{\theta_{y,[i/2]}}{2})$

where $l_{c1,2}$ are vectors representing the lengths of the rigid connectors mounted between the segments of length $l_{s1,2}$.

This kind of geometrical transformation is not considering the shrink in length that happens during bending. To take into account this problem we can use:

$$l_s = 2 l_0 \times \frac{\sin(\frac{\theta}{2})}{\theta} \quad \text{for } \theta \neq 0 \quad (9)$$

However, such a computation would first need us to recover θ variable, introducing new singularities. It is therefore suggested to use only a constant scaling factor online, since approximations in the *MPC* don't lead to error accumulation and wind-up, while considering the complete formula for offline computations only.

We have implemented a *Robust Tube-based MPC* [8], computing the feedback gain matrix K according to the Discrete-time Algebraic Riccati Equation [16]. The need for a robust version of MPC instead of a standard one is due to the large model mismatch and approximations, that have been considered as disturbances over the state evolution.

What is theoretically an offline computation, in our codebase needs to be computed online since we have different state-space matrices each control loop, when we update the state. This represents a relevant issue in terms of computation time, adding a lot of overhead. However, having selected a very conservative control rate allowed us to rarely miss the real time synchronization deadlines. The second step needed for a *Robust-MPC* implementation is to compute the minimum invariant set \mathcal{E} under the terminal LQR control law, but our codebase would force us to do that online each iterations, something unfeasible due to the complexity of the computations.

We were therefore required to deal with noise bounds in an alternative way to prevent the system from being too conservative. We chose to apply the constraint tightening after the optimization, via a function that maps the output of $K(x_i - z_i)$ into a set that fulfills input constraints. But using this solution made us to lose all convergence guarantees from DARE, so in the future different approaches will be investigated.

C. Robust MPC formulation

In eq. (10) is reported our *Robust MPC* scheme. We have implemented it using the CasADi C++ library [17] interfaced with IPOPT solver [18]. This formulation is aiming for simple trajectory following tasks and has been tested successfully on various different settings.

$$J^*(q(k), \dot{q}(k)) = \min_{q, \dot{q}, u} \mathcal{L} \quad (10)$$

s.t. dynamics $u \in \mathcal{U}$

$(q(N), \dot{q}(N)) \in \mathcal{X}_f \quad (q(0), \dot{q}(0)) \in \mathcal{X}_0$

The cost function in eq. (11) is defined with delta-formulation, so minimizing the distance between the reference trajectory *ref*, provided as external input parameter, and the end-effector symbolic transformation E_q computed according to eq. (8). It is possible to include a multidimensional trajectory in the optimization problem to feed the *MPC* with future references, so that the optimizer is able to take them into account during planning. This option is a unique advantage of *MPC* over other controllers, and it is extremely useful when it is needed to track fast-varying references with abrupt variations. In the stage cost we have regularized both \dot{q} curvature speed and the pressure input u , since the optimization is affected by high variance. We included also a penalization on the pressure variation, preventing the system from often choosing one of the two input limits. Stage cost and terminal cost are both defined as quadratic functions with

positive definite matrices.

$$\begin{aligned} \mathcal{L} = & (E_q(N) - ref(N))^T Q_N (E_q(N) - ref(N)) + \\ & + \sum_{k=0}^{N-1} (E_q(k) - ref(k))^T Q (E_q(k) - ref(k)) + \\ & \dot{q}(k)^T S \dot{q}(k) + u(k)^T R u(k) + \Delta u(k)^T R \Delta u(k) \end{aligned} \quad (11)$$

The state-space dynamic equation has been obtained according to eq. (7), resulting in:

$$\begin{bmatrix} q(k+1) \\ \dot{q}(k+1) \end{bmatrix} = A_d \begin{bmatrix} q(k) \\ \dot{q}(k) \end{bmatrix} + B_d u(k) + W_d \quad \forall 0 \leq k < N \quad (12)$$

For the experiments we have decided to not consider pressure dynamics to keep the computational effort light enough, however [12] have shown as an augmented state-space helps in obtaining better results. In order to implement their dynamic constraint as eq. (13), it is needed to define the pressure dynamic evolution, possibly empirically, and properly adapt the cost function.

$$\begin{bmatrix} \dot{q}(k+1) \\ q(k+1) \\ P_0(k+1) \\ P_1(k+1) \end{bmatrix} = \bar{A}_d \begin{bmatrix} \dot{q}(k) \\ q(k) \\ P_0(k) \\ P_1(k) \end{bmatrix} + \bar{B}_d \begin{bmatrix} P_{0,d}(k) \\ P_{1,d}(k) \end{bmatrix} \quad (13)$$

In eq. (13) $P_{0,1}$ are the 2 chambers pressures and P_d the commanded pressures.

Pressure constraints in eq. (14) model absolute limits and step variation Δu . In this way we can account for actuator internal constraints during optimization. However, since our model is linearized about the initial state, extending the constraining along the whole prediction horizon could lead to unfeasibilities. In fact we cannot capture stiffness and damping changing during the virtual state evolution since we update matrices only at the beginning of the control loop; allowing the system to produce any pressures towards the end of the horizon would limit the problems related to our static model and doesn't provide any danger since the only applied input is the first one, where limits are present. However, feasibility guarantees of the *MPC* formulation are lost and our controller could fail unexpectedly during the optimization, so we needed to properly account for it.

$$\begin{aligned} \mathcal{U} : \quad & -\Delta u < u(0) - u_{old} < \Delta u \quad \wedge \\ & -\Delta u < u(k+1) - u(k) < \Delta u \quad \wedge \\ & p_{min} < u(k) < p_{max} \quad \forall 0 \leq k < N/4 \end{aligned} \quad (14)$$

The terminal constraint has been object of specific care, getting to use a safety filter inspired method.

$$\mathcal{X}_f : \quad \dot{q}(N) = 0 \quad (15)$$

Predictive safety filters have been introduced in reinforcement learning setting, when policy training is executed directly on the real hardware. Since the goal is to explore enough the state-space to collect valuable information, it is very likely that the system would get in dangerous situations, given a not perfectly known state evolution and unpredictable effects of new actions on unexplored domains. The idea of the safety filter is to continuously plan a trajectory able to

bring back the system into a safe condition and adapt control inputs from the learning algorithm to ensure it. Refer to [19] for more information about safe RL.

With SoPra we could have possible problems related to unwanted oscillations derived from the model uncertainties at high speeds. Considering a *MPC* trajectory that, while minimizing the euclidean distance from the reference, ensures the existence of a control sequence to bring back the robot to a stationary condition turned out to be extremely valuable against model approximations. Particularly, this choice helps in limiting too fast pressure variations and, keeping curvature speed controlled, makes the unwanted effects due to Coriolis term discrepancy almost negligible. However, a horizon long enough has to be chosen, otherwise the robot would be too limited and would be barely able to follow any trajectories.

The initial constraint is designed as required for the *Robust MPC*:

$$\mathcal{X}_0 : \quad A_I q(0) < Q_0 \quad A_I \dot{q}(0) < \dot{Q}_0 \quad (16)$$

Q_0 and \dot{Q}_0 represent the offsets of a rectangular polytope centered around $q(k)$ and $\dot{q}(k)$ measured values at time $k=0$. According to the standard *Tube MPC* [8], the control input is:

$$p^* = u(0) + K \begin{bmatrix} q(k) - q(0) \\ \dot{q}(k) - \dot{q}(0) \end{bmatrix} \quad (17)$$

exploiting the initialization within a neighboring set. Since we didn't tightened the constraints on pressure, being too complex online or too restrictive offline, we have then limited the amount of variation of p^* from nominal input $u(0)$ at actuation phase.

The optimal problem described in eq. (10) is modular: it can be adjusted with additional elements to address more complex settings. As a first step, we have implemented two different ways to account for obstacles, each of them with peculiar capabilities.

The first method penalizes those states that lead to end-effector configurations too close to specified obstacles. We add the cost term shown in c18 with a constant penalization $L > 0$, that smoothly decreases with distance from obstacles o_i . We have used an exponential function since easily differentiable during problem solving, with l factor to weight distance variation. However, working on-line, it cannot be used for many obstacles without increasing too much the computation time and therefore losing tracking performance. It represents a straightforward implementation to avoid unexpected sparse obstacles, but cannot be used systematically and further could affect the controller convergence.

$$\mathcal{L} + = \sum_{k=0}^{N-1} \sum_{i=0}^{N_{obs}} \frac{L}{\exp(l(E_q(k) - o_i)^T (E_q(k) - o_i))} \quad (18)$$

The second method can account for many obstacles at the same time, since the algorithm that we derived from [7] works offline. We compute a set $\mathcal{X} : \{q \mid A_I q < b_I\}$ for the joint variables and then design a *soft-MPC* based on it, as

summarized in eq. (19).

$$\begin{aligned} \min_{q, \dot{q}, u} \mathcal{L} + \sum_{k=0}^{N-1} \varepsilon(k)^T E \varepsilon(k) \quad (19) \\ \text{s.t. dynamics} \\ A_I q(k) < b_I + \varepsilon(k) \quad \forall 0 \leq k < N \\ u \in \mathcal{U} \quad (q(N), \dot{q}(N)) \in \mathcal{X}_f \quad (q(0), \dot{q}(0)) \in \mathcal{X}_0 \end{aligned}$$

Soft-MPC is a *MPC* formulation where some of the constraints are relaxed via the addition of a slack variable ε , properly penalized in the cost. With this method is possible to operate under constraint configurations without the risk of incurring into unfeasibilities. However, only non-critical constraints can be softened, since failing to meet pressure requirements could lead to hardware damage.

Our implementation provides a restriction at joint-level, so multiple obstacles and further constraints can be managed. The algorithm is briefly summarized in algorithm 1. The function *check_inclusion*(q_l, q_u) randomly tries different curvature combinations in the interval provided and checks whether obstacles are hit and targets are reached. If most of targets can be reached within a neighborhood while avoiding obstacles, then returns *True* for the proposed interval and the previous optimal solution is updated checking the 2-norm set. This approach allows to consider infinitely many constraints at each position along the arm, but can provide only pseudo-symmetric sets. Therefore, sparse obstacles cannot be handled efficiently, while it is really suited for more homogeneous restrictions of the state-space. Following [7] it is then possible to include further geometry and actuation based constraints, for a more aware control law.

Algorithm 1 Offline computation of state constraints.

```

 $q_l \leftarrow [-\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}] \quad q_u \leftarrow [\frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}]$ 
if check_inclusion( $q_l, q_u$ ) then
    Use standard limits
else
     $q_l^* \leftarrow [0, 0, 0, 0] \quad q_u^* \leftarrow [0, 0, 0, 0]$ 
    while  $i < N_{\text{trials}}$  do
         $q_l^t \leftarrow \text{Uniform\_Distribution}(q_l, q_u)$ 
         $q_u^t \leftarrow \text{Uniform\_Distribution}(q_l^t, q_u)$ 
        if  $(\|q_u^t - q_l^t\|_2 > \|q_u^* - q_l^*\|_2) \wedge$ 
        check_inclusion( $q_l^t, q_u^t$ ) then
            New solution found
             $q_l^* \leftarrow q_u^t \quad q_u^* \leftarrow q_l^t$ 
        end if
         $i \leftarrow i + 1$ 
    end while
end if

```

III. SIMULATION

We have run intensive simulations to validate our new method. The simulator is custom-made, working on a URDF model of the arm. Drake [15] has been mainly used to solve the dynamics. As benchmark, we have considered

the quasi-static controller, an approach similar to the one used in previous works for task-space references. It is based on inverse kinematics (no dynamic information) and small iterative updates to reduce the error; it is therefore slow and needs double the time of our MPC to follow the same trajectory, while running at the same rate.

In fig. 3 we have collected results of our *Robust MPC* when performing a circular tracking while dealing with an obstacle at end-effector level. The penalized *MPC* (option 1, in blue) performs ideally far from the obstacle and reacts to it only when within the neighborhood tuned in eq. (18). By adapting the penalization term, we can achieve larger or narrower turns around the obstacle, assumed symmetric. However, the same approach would work also for obstacles of more complex shapes, by penalizing all the corner positions. The *Soft-MPC* of eq. (19), red in the plots, instead delivers a more conservative output, where many more configurations are prevented from being reached. This difference is due to the nature of the offline problem we are solving, that isn't able neither to provide complex shapes nor to fully optimize the final set. These issues, that will be addressed in the future, are particularly evident for sparse obstacles configurations as the one analysed in fig. 4. We can see how the *Robust MPC* (on the left), that achieves an almost perfect tracking, can be adapted to account for obstacles, but the two options we provide behave differently. The penalized *MPC* avoids areas of large cost, whose positions correspond to the obstacles and are defined either offline or online, while the *Soft-MPC* has offline optimized over a joint variable set and works on it, even though could be extremely conservative for sparse obstacles configurations (as for the obstacle in position $(-0.14, 0)\text{m}$), or possibly not effective (as happens for obstacle centered in $(0, 0.14)\text{m}$). In fig. 4 we can also observe how the quasi-static benchmark, despite using double the time, is still too slow to properly reach the required setpoint, since no dynamical information is available and therefore only almost static motions are possible.

The *Soft-MPC* with offline constraining has been designed to be applied in conditions of homogeneous restriction of the task space, for example a narrow hole along the length of the arm. This problem could represent a realistic setting where the robot needs to explore cavities and therefore being restricted in its motion capabilities. With our algorithm, joint configurations are limited to the only ones that allow enough motion at end-effector, while keeping almost constant the intermediate position. The results are shown in the video supporting this submission. Furthermore, using a more complex computation that features also mechanical information as in [7], resulting constraints would be even more appropriate.

IV. PHYSICAL EXPERIMENTS

We have tested our control algorithm on a real robotic arm, SoPrA [1]. Instead of pursuing the best performance, we preferred to focus on showing the effectiveness of the concepts we have validated in simulation in a real environment, where model approximations and simplifying assumptions could have affected the control. In fig. 5 the experimental set-up

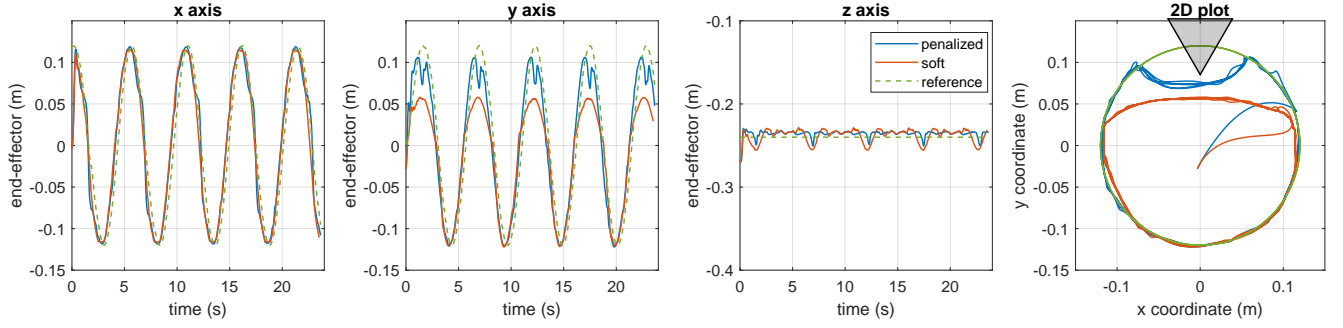


Fig. 3: *Robust MPC* simulation results. The penalized *MPC* is shown in blue and the *Soft-MPC* is shown in red. For sparse obstacles, the off-line computed set is more restrictive than the online penalization. The arm controller should follow the circular reference, but the presence of obstacles forces it to adapt.

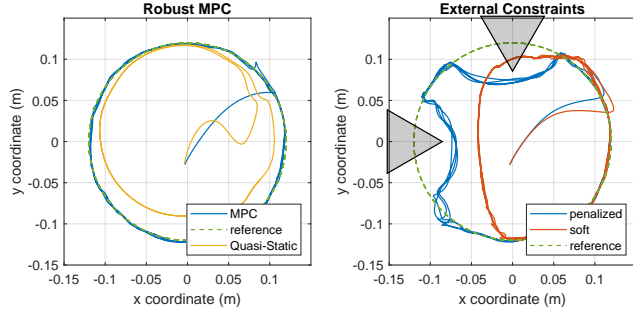


Fig. 4: Simulation of circular trajectories. Left: given no additional constraints, tracking accuracy is high. A comparison with a quasi-static controller is provided. Right: a setting with sparse constraints is presented. The *Soft-MPC* with offline computed set doesn't behave optimally under these conditions, since it is partially hitting the virtual obstacle due to its *soft* implementation.

has been summarized. Important to notice that the feedback is provided via a motion capture system made by infrared and color cameras. Reflective markers are attached in three positions along the arm: the base, the middle connector and the end-effector. The commanded pressures are fed to a proportional valve array that actuates the robot.

Our controller runs at 15 Hz, due to the limitations of the hardware used, with horizon length of 7 steps (the *MPC* therefore predicts 0.5 seconds in the future). Control rate could be increased to almost 30 Hz by implementing multi-threading and possibly further by selecting more efficient solvers. Throughout our experiments we have seen a consistent performance improvement with frequency increasing, so we believe that reaching a 50 Hz control rate will allow this implementation to match other controllers' tracking accuracy. Although the low rate makes our controller prone to instability and with a low noise-rejection ability, we observe promising results thanks to the potential of the *MPC* formulation in conditions where most other controllers would have failed. This implementation can therefore be applied in a vast range of situations where only cheap sensors are available, since it can work without the need of a high feedback rate.

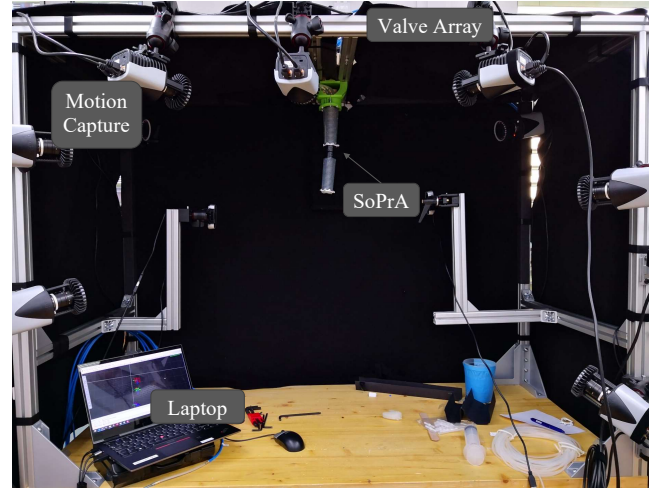


Fig. 5: The SoPrA arm used consists of two fiber-reinforced segments made by three positive pressure chambers each. Between soft segments rigid connectors are present, and tubes are wired through them. A proportional valve array, placed above the arm, is used to independently pressurize the six cavities. The motion capture measures the segments' curvature.

The optimization variables q, \dot{q} are initialized for the whole horizon with the measured value each iteration, while u is initialized at zero. All dynamic parameters for eq. (12) are numerically computed and provided to the solver each control loop.

In fig. 6 are reported the ideal results of our controller performing two turns of 10 cm radius in 25 seconds. We observe a delay due to the pressure actuation that hasn't been addressed during the modeling phase. However, our implementation would allow to consider it with an approach similar to eq. (13), where pressure dynamics is directly modeled. The future reference knowledge of *MPC* would allow to account for the actuation delay and properly follow the trajectory variations, as no-other controller can do. On the 2D plot we observe a good tracking performance, with the controller able to damp out most of oscillations that occur due to the model mismatches and mechanical inhomogeneity.

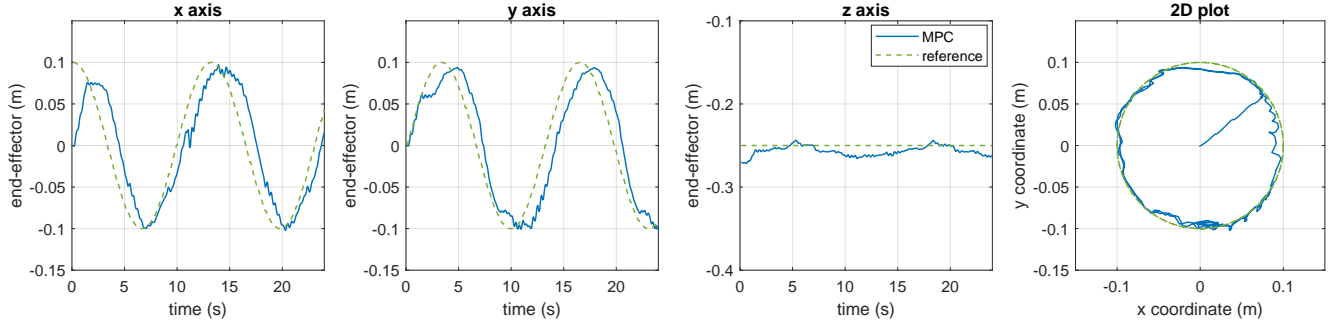


Fig. 6: Real-world deployment of *Robust MPC* on SoPrA. Controller performed two circular end-effector motions with a 10 cm radius. We observe a constant delay due to pressure input, but the tracking remains accurate. Controller runs at 15Hz.

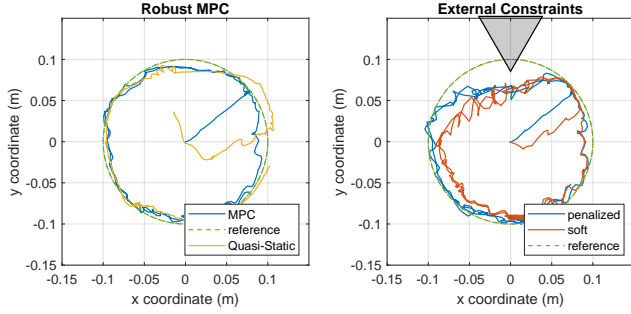


Fig. 7: Real experiment showing two circular end-effector rotations in 20 seconds. Left: *Robust MPC* without additional constraints. Right: penalized *MPC* and *Soft-MPC* with constrained set. Given this sparse obstacle setting, *Soft-MPC* performs worse, but still achieves its goal accounting for external constraints. A benchmark for left plot is provided.

In fig. 7 we show a more challenging setting, with 10 seconds turns in presence of obstacles. Both our proposed approaches, even though affecting ideal performances, are able to deal with the more complex environment as expected from the simulation. We can however observe one drawback of the offline set computation, that prevents the arm from reaching the required position also where no obstacles are present; furthermore, we also observe a more oscillating behavior. These problems are due to the approach we chose in computing the set, that can constrain excessively the curvature variables. Checking the ideal performance only, the MPC formulation is comparable to quasi-static in terms of noise rejection, but allows for double the speed in reference tracking. Furthermore, MPC formulation can be used for additional tasks as shown in this paper.

One interesting experiment is provided in fig. 8. Here, the commanded reference is a square lying outside of the reachability of SoPrA, that has a spherical task-space. Despite the inaccurate tracking performances, these results testify a few advantages that *MPC* has over other controllers, extremely important to achieve realistic applications. First, our optimal controller is able to follow an unreachable reference just minimizing the error, without the need of reducing it to 0 (something that would happen with an integral term). Secondly, corners (abrupt reference variations) are never

missed due to the future knowledge. This result is important for possible applications since the controller is more aware about its surroundings and can act accordingly. Lastly, even in presence of quite large model uncertainties, our robust formulation is able to keep the arm almost steady in position control, possibly with a bias. This result is expected from *MPC* theory, and many ways to reduce the issue exist.

According to our simulated and physical results, we can certify the effectiveness of our approach, that allow us to include internal constraints for actuation limits and pressure dynamics, external constraints from the environment and admissible configurations dependent on shape and manufacturing in the same modular framework that makes easy to enable new features according to the task requirements. We furthermore believe that *MPC* would also allow to control underactuated robots with good performances; we achieved preliminary results towards this direction but we couldn't test intensively due to hardware and software limitations.

V. CONCLUSION & FUTURE WORK

We have shown how *Model Predictive Control* can be adapted to soft robotics task-space control. *MPC* is proposed as a way to handle many of the challenges typical of this control domain, such as actuation limits and reachable task-space limitations. We have built a framework based on the *Augmented Rigid Body Model* [2], [3] that can be used to control soft robotic arms also in prohibitive settings of low control rate and large model deviations. Our unified and modular approach allows to address many different issues at the same time, makes extremely easy to add new features and can be directly applied to arms of different shapes and actuation as long as an efficient kinematic representation can be retrieved.

This work represents a step towards new achievements in soft robotic arms control for practical applications, since controllers based on our method would allow the use of these robots in a realistic and cooperative environment, where safety and constraints' meeting are of primary importance.

Remaining issues to be addressed in future work are related to the large computation time and the model approximations; these have to be handled in order to increase the robustness in more challenging environments. To speed-up the control process many technical improvements can be

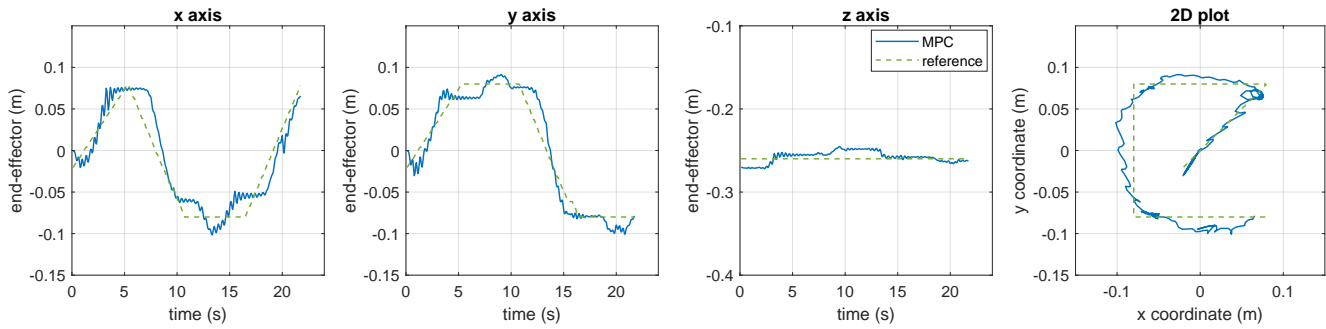


Fig. 8: Square trajectory at constant height. SoPrA cannot reach the commanded positions, but *MPC* can deal with the limits trying to deliver a trajectory that minimizes the error. Since the future trajectory is known, no corner is overshoot.

employed, such as multi-threading or more advanced solvers. To face model approximations, our dynamics can be enriched with a data-driven oracle since the underlying *MPC* framework makes it easy to integrate recently proposed learning methods as done in [20], [21]. The proposed framework would allow any future research to exponentially expand its applications, since better and more aware controllers might be designed, suitable for complex and potentially dangerous tasks that now are domain of rigid robotics only. Another long term goal is to increase the planning horizon of our *MPC*; with this improvement our controller will be able to command faster motions by taking advantage of the stored potential energy when bending the silicon elastomer of soft robots, a unique capability of *MPC* framework.

ACKNOWLEDGMENT

We thank Yasunori Toshimitsu and Oliver Fischer for their support, particularly their C++ codebase used for previous control approaches upon which we built this *MPC* control framework. We also thank Amirhossein Kazemipour for his important insights during the planning phase of this work.

REFERENCES

- [1] Y. Toshimitsu, K. W. Wong, T. Buchner, and R. Katzschmann, "SoPrA: Fabrication & Dynamical Modeling of a Scalable Soft Continuum Robotic Arm with Integrated Proprioceptive Sensing," 2021. [Online]. Available: <http://arxiv.org/abs/2103.10726>
- [2] R. K. Katzschmann, C. D. Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, "Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model," *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, pp. 454–461, 2019.
- [3] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 490–513, 2020.
- [4] R. K. Katzschmann, A. D. Marchese, and D. Rus, "Autonomous object manipulation using a soft planar grasping manipulator," *Soft Robotics*, vol. 2, no. 4, pp. 155–164, 2015.
- [5] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Dynamic control of soft robots interacting with the environment," *2018 IEEE International Conference on Soft Robotics, RoboSoft 2018*, pp. 46–53, 2018.
- [6] A. Kazemipour, O. Fischer, Y. Toshimitsu, K. W. Wong, and R. K. Katzschmann, "A robust adaptive approach to dynamic control of soft continuum manipulators," *arXiv preprint arXiv:2109.11388*, 2021.
- [7] C. Della Santina, A. Bicchi, and D. Rus, "Dynamic Control of Soft Robots with Internal Constraints in the Presence of Obstacles," *IEEE International Conference on Intelligent Robots and Systems*, pp. 6622–6629, 2019.
- [8] B. T. Lopez, J. P. Howl, and J. J. E. Slotine, "Dynamic tube MPC for nonlinear systems," *Proceedings of the American Control Conference*, vol. 2019-July, pp. 1655–1662, 2019.
- [9] M. J. Powell, E. A. Cousineau, and A. D. Ames, "Model predictive control of underactuated bipedal robotic walking," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, pp. 5121–5126, 2015.
- [10] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pamc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.
- [11] L. Hewing, K. P. Wabersich, and M. N. Zeilinger, "Recursively feasible stochastic model predictive control using indirect feedback," 2019.
- [12] C. M. Best, M. T. Gillespie, P. Hyatt, M. D. Killpack, L. Rupert, and V. Sherrod, "Model Predictive Control for Pneumatically Actuated Soft Robots," *IEEE Robotics & Automation Magazine*, vol. 3, no. 9, p. 31, 2016.
- [13] P. E. Hyatt, "Robust real-time model predictive control for high degree of freedom soft robots," 2020. [Online]. Available: <https://scholarsarchive.byu.edu/etd/8453>
- [14] C. M. Best, L. Rupert, and M. D. Killpack, "Comparing model-based control methods for simultaneous stiffness and position control of inflatable soft robots," *International Journal of Robotics Research*, vol. 40, no. 1, pp. 470–493, 2021.
- [15] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: <https://drake.mit.edu>
- [16] V. Kučera, "The discrete riccati equation of optimal control," *Kybernetika*, vol. 8, no. 5, pp. 430–447, 1972.
- [17] J. A. 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. [Online]. Available: <https://doi.org/10.1007/s12532-018-0139-4>
- [18] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, 2006.
- [19] J. García, Fern, and o Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015. [Online]. Available: <http://jmlr.org/papers/v16/garcia15a.html>
- [20] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," *2018 IEEE International Conference on Soft Robotics, RoboSoft 2018*, pp. 39–45, 2018.
- [21] P. Hyatt, C. C. Johnson, and M. D. Killpack, "Model reference predictive adaptive control for large-scale soft robots," *Frontiers in Robotics and AI*, vol. 7, p. 558027, 2020.