Model Reference Predictive Adaptive Control for Large Scale Soft Robots

Phillip Hyatt[1], Curtis Johnson[1], and Marc D. Killpack[1]

***Abstract*— Past work has shown Model Predictive Control (MPC) to be an effective strategy for controlling continuum joint soft robots using rudimentary models, however the inaccuracies of these models often mean that an integration scheme must be combined with MPC. Presented in this work is a novel dynamic model formulation for continuum joint soft robots which is more accurate than the authors' previous models yet remains fast enough for MPC. This model is based on the Piecewise Constant Curvature (PCC) assumption and a relatively new configuration representation and allows for computationally efficient simulation. Due to the difficulty in determining model parameters (damping and spring effects) as well as effects common in continuum joint soft robots (hysteresis, complex pressure dynamics, etc.), we submit that most model-based controllers of continuum joint soft robots would benefit from some form of model adaptation. In this work a novel form of adaptive MPC is presented based on Model Reference Adaptive Control (MRAC). We call our novel adaptive MPC approach MRPAC. We show that like MRAC, MRPAC is able to compensate for "known unknowns" such as unknown inertias and spring and damper coefficients. Our experiments also show that like MPC, MRPAC is also robust to "unknown unknowns" such as unmodeled external forces and any forces not represented in the form of the adaptive model. Experiments in simulation and hardware show that MRPAC outperforms MPC and MRAC in every case except for that in which MPC uses a perfect model in simulation.**

I. Introduction

Large scale soft robots hold promise as platforms which are safe to work in human and delicate environments and which can accomplish tasks for which rigid robots are ill suited. Some tasks for which large scale soft robots are uniquely well suited include whole-arm wiping tasks, reaching through unmodeled cluttered environments, and any task where incidental unmodeled contact is likely or desirable. Continuum joint soft robots have specifically been modeled after examples in nature which excel at these types of tasks (anteaters, octopi, elephants, etc.).

One major obstacle to the use of continuum joint soft robots is the accuracy of their control. Because soft robot continuum joints are not necessarily constrained to rotate about a single well defined axis, even the kinematic modeling of these robots is relatively difficult when compared to rigid robots. Furthermore, the rigid body dynamics which govern the motion of traditional robots are complicated in continuum joint soft robots by pressure dynamics, energy storage and dissipation in the joints, as well as buckling in some load cases. These factors make the accurate modeling and model-based control of continuum joint soft robots very difficult.

In this work we present a novel method for dynamic modeling of continuum joint robots which can be evaluated fast enough for real-time Model Predictive Control (MPC). This novel dynamics model is in fact a small extension of well-established dynamics models of continuum joint robots based on piecewise constant curvature (PCC) approximations, and a relatively new choice of configuration variables.

We also present a novel form of adaptive MPC which can adapt this model in order to improve dynamic performance and eliminate steady state error. The adaptive law and much of the theoretical basis for this controller are derived from Model Reference Adaptive Control techniques.

The structure of this paper is as follows: Section II will discuss the state of the art in continuum soft robot modeling and control, as well as the hardware, models and methods specific to this work; Section III will explain the hypotheses about the model and proposed controller as well as the design of the experiments performed; Section IV will report the results of the experiments performed and discuss their importance; Section V will discuss the importance of the presented work to the field and provide suggestions for future work.

II. Materials and Methods

A. Related Work

There is a significant body of work which has been done to accurately model the kinematics and dynamics of soft robots. In [1], [2] the continuum joint is modeled using Cosserat-beam theory. In [3] and [4] methods based on recursive Newton Euler approaches are taken, while in [5]

[1]All authors are with Brigham Young University, Mechanical Engineering Department

and [6] dynamic equations are derived using Lagrangian mechanics. In [7] and [8] lumped parameter models are derived by dividing the continuum joint into a number of finite length sections. The trade-off between accuracy and computational complexity in these methods can be seen by varying the number of the finite sections. [9] provides a more comprehensive review of dynamics models for soft and continuum joint robots. Notably, there has also been work to show that learned models can represent soft robot dynamics as in [10].

In [11] and [12] the authors derive the dynamic equations of a continuum arm by integrating over infinitesimal discs and using the method of Lagrange. No assumptions of constant curvature are made. These works are similar to our own, the main differences being our choices of generalized coordinates and our assumption of constant curvature. These two differences allow us to derive closed form analytical expressions for the terms in our equations of motion such as the mass and coriolis matrices.

In [13] and [14] the authors derive simpler models based on the PCC assumption. However they neglect generalized forces caused by rotations (inertias). They also model the mass of each PCC section as being concentrated at a point which is fixed in some coordinate frame. Because the mass and inertia of the joints used in this work are non-negligible, we model the mass as distributed uniformly throughout infinitesimal discs and the center of mass of each section is calculated analytically assuming uniform density.

Control strategies for soft robots vary from open-loop control such as in [15], [16] to Reinforcement Learning [17] to MPC [18]. In [19] and [20] the authors demonstrate the performance of MPC on the same joints used for this work. These implementations of MPC used a learned model of the dynamics based on a less-accurate representation of the continuum joint dynamics. The model inaccuracy involved in that work prompted the development of the more accurate model and adaptive control techniques presented in this work.

Given a dynamic model of the correct form, the nature of soft robots is still such that certain parameters of that model may be difficult to estimate. Adaptive MPC is a well established method in the literature which combines the strengths of MPC with adaptive control [21], [22]. The method developed in this paper is a form of adaptive MPC which borrows ideas from Model Reference Adaptive Control for manipulators [23]. Specifically, our work can be considered

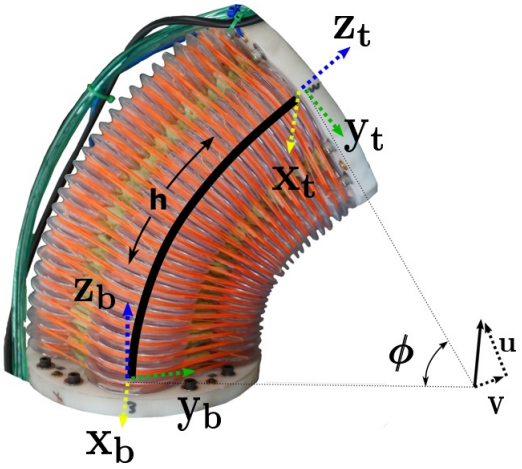

Fig. 1: A continuum joint such as the ones that comprise the robots used for this work.

an extension to that in [24]. The main extensions are that the method in this work uses a single MPC controller for all joints as opposed to one controller for each joint. The regressor used for parameter estimation is also refined based on our new model and the adaptation scheme is altered to consider both position and velocity errors.

B. Robot Platform Description and Modelling

The robots used for this work are composed of continuum joints such as the one seen in Figure 1. These joints are made of four separate pressure controlled chambers surrounding a relatively in-extensible spine, allowing each joint to bend about two axes. We choose to model the kinematics of each joint using arcs of constant curvature. Each arc, which traces out the path in space occupied by the inextensible spine, can be described using three variables as described in [25]. These variables are the length of the in-extensible spine (h) and the two components of the vector which describes the rotation from the bottom to the top of the arc (u and v). These values are labeled in Figure 1. We assume that the spine is perfectly in-extensible so h becomes a constant kinematic parameter.

First we note some useful kinematic relationships. The variables u and v can be thought of as the nonzero portions of the axis-angle representation of the rotation between the bottom and top discs of the joint and therefore

$$\phi = \sqrt{u^2 + v^2} \tag{1}$$

where ϕ is the magnitude of the axis-angle vector $[u, v, 0]^T$. Note that because the frame tangent to the arc rotates as the length along the curve l is increased, we know that ϕ, u, and v are not constant along the entire arc. However we note

that the vector ρ from the base of the joint to the center of curvature is the same for all points along the arc. At any point l along the arc this value can be calculated as

$$\rho = \frac{l}{\phi^2} \begin{bmatrix} v \\ -u \\ 0 \end{bmatrix}. \tag{2}$$

Because the magnitude of this vector $||\rho||$ is the radius of curvature, we may also relate ϕ and l using the arc-length formula

$$\phi = \frac{l}{||\rho||} \tag{3}$$

We now wish to derive a means by which we can calculate u and v at any point l along the arc given only l, h, and u and v at the end of the arc. Given a point which lies at a distance l along the arc, we may say using Equation 2

$$\begin{aligned} \rho_l &= \rho_h \\ \frac{l}{\phi_l^2} \begin{bmatrix} v_l \\ -u_l \\ 0 \end{bmatrix} &= \frac{h}{\phi_h^2} \begin{bmatrix} v_h \\ -u_h \\ 0 \end{bmatrix}. \end{aligned} \tag{4}$$

Replacing ϕ terms using Equation 3 we obtain

$$\begin{aligned} \frac{l||\rho||^2}{l^2} \begin{bmatrix} v_l \\ -u_l \\ 0 \end{bmatrix} &= \frac{h||\rho||^2}{h^2} \begin{bmatrix} v_h \\ -u_h \\ 0 \end{bmatrix} \\ \begin{bmatrix} v_l \\ u_l \\ 0 \end{bmatrix} &= \frac{l}{h} \begin{bmatrix} v_h \\ u_h \\ 0 \end{bmatrix}. \end{aligned} \tag{5}$$

In other words, the generalized coordinates u and v vary linearly along the length of the arc. This will become a very useful property of this kinematic representation when deriving equations of motion.

Using the method of Lagrange, the equations of motion for a system of rigid bodies take the form

$$M\ddot{q} + C\dot{q} + g = \tau \tag{6}$$

where M is the mass matrix, C is the Coriolis matrix, g is a vector of gravity torques, q is a vector of the generalized coordinates, and τ is a vector of the generalized torques. These matrices are derived using partial derivatives of kinetic and potential energy terms. Since partial derivatives are easily taken using a symbolic mathematics toolbox such as sympy, the problem of dynamic modeling is reduced to the selection of generalized coordinates and the representation of kinetic and potential energy.

In order to accurately express kinetic and potential energy we choose to model the continuum joint, as many have done before, with an infinite set of infinitesimally small discs. However the assumption of constant curvature, the choice of generalized coordinates, and current tools in symbolic math libraries will allows us to produce analytical expressions for M, C, and g, whereas previous methods have not yielded these closed form expressions.

We can define the kinetic energy of an infinitesimally thin disc as

$$\begin{aligned} T &= \frac{1}{2}(\mu dl)\dot{p}^T\dot{p} + \frac{1}{2}\omega^T I \omega \\ &= \frac{1}{2}(\mu dl)\dot{p}^T\dot{p} + \frac{1}{2}\omega^T \begin{bmatrix} \frac{\mu dl r^2}{4} & 0 & 0 \\ 0 & \frac{\mu dl r^2}{4} & 0 \\ 0 & 0 & \frac{\mu dl r^2}{2} \end{bmatrix} \omega \\ &= \frac{1}{2}(\mu dl)\dot{p}^T\dot{p} + \frac{1}{2}\left[\frac{\mu dl r^2}{4}\omega_x^2 + \frac{\mu dl r^2}{4}\omega_y^2 + \frac{\mu dl r^2}{2}\omega_z^2\right] \\ &= \frac{\mu}{2}\left[\dot{p}^T\dot{p} + r^2\left(\frac{1}{4}\omega_x^2 + \frac{1}{4}\omega_y^2 + \frac{1}{2}\omega_z^2\right)\right] dl \end{aligned} \tag{7}$$

where μ is the linear density of the disc, dl is some infinitesimal length, $\dot{p}$ is the velocity of the center of the disc, ω is the angular velocity of the disc expressed in the disk frame, and I is the inertia of the disc expressed in the disc frame.

The linear and angular velocity of each disk ($\dot{p}$ and ω) can be found using a configuration dependent jacobian J which is defined such that

$$\begin{aligned} \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} &= J(u, v, l) \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} \\ \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} &= \begin{bmatrix} J_{\dot{p}}(u, v, l) \\ J_{\omega}(u, v, l) \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix}. \end{aligned} \tag{8}$$

Using this relationship, we see that we can simplify the expression for kinetic energy (Equation 7) by scaling portions of the jacobian. The new inertia-weighted jacobian is defined as

$$J_{weighted} = \begin{bmatrix} \sqrt{\mu} J_{\dot{p},x} \\ \sqrt{\mu} J_{\dot{p},y} \\ \sqrt{\mu} J_{\dot{p},z} \\ \frac{\sqrt{\mu} r}{2} J_{\omega,x} \\ \frac{\sqrt{\mu} r}{2} J_{\omega,y} \\ \frac{\sqrt{\mu} r}{\sqrt{2}} J_{\omega,z} \end{bmatrix} \tag{9}$$

allowing us to rewrite Equation 7 for the kinetic energy

of a disc as

$$T = \frac{1}{2}\dot{q}^T J_{weighted}(u,v,l)^T J_{weighted}(u,v,l)\dot{q}dl. \quad (10)$$

By treating a continuum joint as a series of infinitesimal discs and integrating the kinetic energy of each disc along the length of the arc we can write the total kinetic energy of a joint as

$$T = \frac{1}{2}\dot{q}^T \left[\int_0^h J_{weighted}(u,v,l)^T J_{weighted}(u,v,l)dl \right] \dot{q} \quad (11)$$

We note here that the Jacobian can be expressed analytically at every point along the joint as a function of l, the configuration variables u_h and v_h, and the kinematic parameter h using Equation 5. Given this analytical expression for $J_{weighted}$ we can integrate over the definite bounds to get an analytical expression for $J^T_{weighted}J_{weighted}$, which we recognize as the joint space inertia matrix or mass matrix M.

We use a symbolic mathematics library (sympy) to calculate $J^T_{weighted}J_{weighted}$, and and to integrate this expression analytically between the definite bounds 0 and h in order to obtain M. Once M has been obtained symbolically in sympy, it is then relatively straightforward to take partial derivatives in sympy in order to obtain an expression for the Coriolis matrix C using the method outlined in [26].

In order to find the gravity torques (g) we must first find the joint center of mass. By inspection we can see that a joint's center of mass must project down onto the vector ρ, however the vector to the center of mass must also contain some component in the z direction (orthogonal to the space spanned by u and v). We find the components of the CoM vector by again dividing the joint into a series of infinitesimal discs of height dl.

Assuming the joint has uniform density along its length, the portion of the CoM vector along the z axis is given by

$$\bar{z} = \frac{\int_0^h z dV}{\int_0^h dV} \quad (12)$$

Using the trigonometric relationship seen in Figure 2, namely

$$z(l) = ||\rho|| sin(\frac{l}{h}\phi) \quad (13)$$

as well as the volume formula for an infinitesimally thin disc

$$dV = \pi r^2 dl, \quad (14)$$

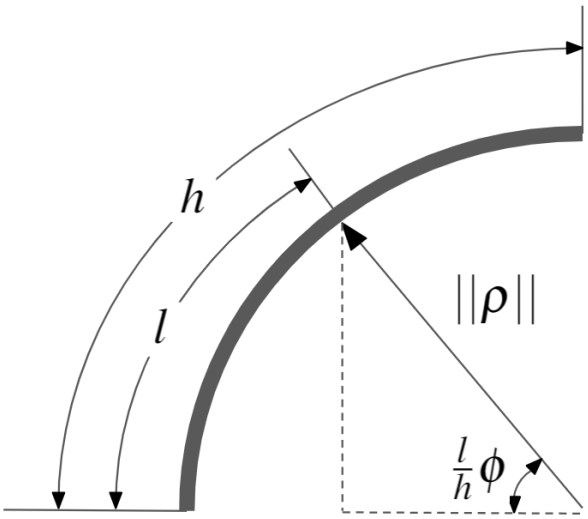

Fig. 2: A side view of a continuum joint

we can now integrate to find $\bar{z}$:

$$\begin{aligned} \bar{z} &= \frac{\int_0^h ||\rho|| sin(\frac{l}{h}\phi)\pi r^2 dh}{\int_0^h \pi r^2 dl} \\ \bar{z} &= \frac{\pi r^2 ||\rho|| \int_0^h sin(\frac{l}{h}\phi)dl}{\pi r^2 h} \\ \bar{z} &= \frac{-\left[||\rho|| \frac{h}{\phi} cos(\frac{l}{h}\phi)\right]_0^h}{h} \\ \bar{z} &= \frac{-||\rho||}{\phi}(cos(\phi) - 1). \end{aligned} \quad (15)$$

Recognizing that $||\rho|| = \frac{h}{\phi}$,

$$\bar{z} = \frac{h}{\phi^2}(1 - cos(\phi)). \quad (16)$$

In order to find the component of the CoM vector which lies in the plane of u and v we follow a similar procedure. We will use x to represent the portion of the CoM vector which lies along ρ. Using the trigonometric relationship seen in Figure 2, namely

$$x(l) = ||\rho||(1 - cos(\frac{l}{h}\phi)), \quad (17)$$

we can now integrate to find $\bar{x}$:

$$\begin{aligned} \bar{x} &= \frac{\int_0^h ||\rho||(1 - cos(\frac{l}{h}\phi))\pi r^2 dh}{\int_0^h \pi r^2 dl} \\ \bar{x} &= \frac{\pi r^2 ||\rho|| \int_0^h (1 - cos(\frac{l}{h}\phi))dl}{\pi r^2 h} \\ \bar{x} &= \frac{||\rho||(l - \frac{h}{\phi} sin(\frac{l}{h}\phi)\big]_0^h}{h} \\ \bar{x} &= \frac{||\rho||}{\phi}(\phi - sin(\phi)). \end{aligned} \quad (18)$$

Recognizing that $||\rho|| = \frac{h}{\phi}$,

$$\bar{x} = \frac{h}{\phi^2}(\phi - sin(\phi)). \tag{19}$$

Using the derived equations for $\bar{z}$, $\bar{x}$, and the normalized version of ρ we obtain the vector from the base of the joint to the center of mass:

$$CoM = \frac{h}{\phi^2}\begin{bmatrix}(\phi - sin(\phi))\frac{v}{\phi}\\(\phi - sin(\phi))\frac{-u}{\phi}\\(1 - cos(\phi))\end{bmatrix}. \tag{20}$$

The potential energy of the joint is simply the dot product of this vector, expressed in the inertial frame, with the gravity vector expressed in the same frame:

$$V = CoM \cdot Grav. \tag{21}$$

Having calculated the potential energy, the gravity torques are calculated simply by taking the negative partial derivative of V with respect to q:

$$g = \frac{\partial V}{\partial q}. \tag{22}$$

The method above has yielded us analytical expressions for M, C, and g with the generalized coordinates u and v. Although complex, these closed form expressions can be exported from sympy into C code which can be evaluated within microseconds, allowing for real-time model-based control of these continuum joints.

C. Development of Model Reference Predictive Adaptive Control

In this section we give brief overviews of both MPC and MRAC in order to clarify notation and establish a background for the development of MRPAC. For in-depth explanations of MPC and MRAC we refer the interested reader to [27] and [28] respectively.

1) Model Predictive Control: Any dynamic system may be represented in state variable form as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w} \tag{23}$$

where $\mathbf{x}$ is the vector of states, $\mathbf{u}$ is the vector of system inputs, and $\mathbf{w}$ is a vector of offsets or disturbances. Using any discretization method (Euler, semi-implicit Euler, matrix exponential, etc.) we can create a discretized state space model:

$$\mathbf{x_{k+1}} = \mathbf{A_d}\mathbf{x_k} + \mathbf{B_d}\mathbf{u_k} + \mathbf{w_d}. \tag{24}$$

The above equation can be used to forward simulate the states of our system, given initial conditions and inputs. In MPC these discretized dynamic equations are the constraints of our optimization while $\mathbf{x_k}$ and $\mathbf{u_k}$ are the optimization variables. In an MPC solver looking forward over a horizon of T time steps, a trajectory optimization may be formulated as:

$$\begin{aligned} J = \sum_{k=0}^{T} \Big[&(\mathbf{x_{goal}} - \mathbf{x_k})^T Q(\mathbf{x_{goal}} - \mathbf{x_k}) \\ &+ (\mathbf{u_{goal}} - \mathbf{u_k})^T R(\mathbf{u_{goal}} - \mathbf{u_k}) \Big] \\ s.t.& \\ \mathbf{x_{k+1}} &= \mathbf{A_d}\mathbf{x_k} + \mathbf{B_d}\mathbf{u_k} + \mathbf{w_d} \quad \forall \quad k = 0, ..., T-1 \end{aligned} \tag{25}$$

where J is the objective function value, $\mathbf{x_{goal}}$ and $\mathbf{u_{goal}}$ are the goal states and inputs respectively. By defining a quadratic cost function and enforcing only linear dynamics constraints we have defined a convex optimization problem suitable for solution using a very fast convex solver. We choose to use the state of the art solver OSQP [29] as the convex solver used for our implementation of MPC. In order to lengthen the horizon of MPC and decrease solve times we also use the input parameterization technique outlined in [27].

MPC solves the above trajectory optimization for the entire horizon of length T, however only the first input ($\mathbf{u_0}$) is applied to the system. After applying this input, the optimization is solved again using state information which is updated using sensor feedback. The discrete-time model can also be updated with a new linearization centered at the new operating point. This process is repeated with MPC only ever applying the first input, but solving over an entire horizon of value T. The fact that MPC is forced to re-solve the trajectory optimization problem with the most current state and model information is what leads to the robustness of MPC to model error as will be shown hereafter.

2) Model Reference Adaptive Control: MRAC is a form of adaptive control which seeks to drive a system to behave like a reference system. Because we are interested in controlling continuum joint soft robots we specifically follow the implementation of MRAC outlined in [23] which is specific to robot manipulators. In this derivation of MRAC for manipulators the authors take advantage of several special properties of manipulator dynamics. Firstly, they express the mass matrix,

coriolis matrix, and gravity torques as being linear in certain manipulator parameters. Stated mathematically:

$$\begin{aligned} M\ddot{q} + C\dot{q} + g &= \tau \\ &= Y(\ddot{q}, \dot{q}, q)a \end{aligned} \tag{26}$$

where $Y(\ddot{q}, \dot{q}, q)$ is a nxp regressor and a is a px1 vector containing the manipulator parameters. In rigid body manipulators it can be shown that a contains the link masses, inertias, and the positions of centers of mass. Using the soft robot continuum joint dynamic model in Section II-B to derive M, C, and g it can be seen by inspection that all of these terms are linear in the joint mass m, as well as square of the joint radius r^2 and joint height h^2.

In [23] the authors present a method by which joint accelerations need not be measured or estimated in order to calculate the regressor. Instead they exploit several properties of manipulator dynamics in order to rewrite the regressor as a function of joint positions, joint velocities, reference system velocities, and reference system accelerations:

$$\tau = Y(q, \dot{q}, \dot{q}_{ref}, \ddot{q}_{ref})a. \tag{27}$$

This is very useful in practice because while accurate measurements or estimates of actual joint accelerations are hard to obtain, the acceleration of the reference system is a calculated value that we know perfectly.

When using MRAC, we generally do not know the parameter vector a perfectly, so we desire to estimate it. We will denote our estimate $\hat{a}$. The adaptive parameter vector $\hat{a}$ is adapted according to the law:

$$\dot{\hat{a}} = -\Gamma^{-1} Y(q, \dot{q}, \dot{q}_{ref}, \ddot{q}_{ref})^T s \tag{28}$$

where

$$\begin{aligned} s &= \dot{\tilde{q}} + \Lambda \tilde{q} \\ \dot{\tilde{q}} &= \dot{q} - \dot{q}_{ref} \\ \tilde{q} &= q - q_{ref}. \end{aligned} \tag{29}$$

The final step in manipulator MRAC as explained in [23] guarantees that not only parameter error, but also position error will be driven to zero. In order to ensure this, the final control law for MRAC is defined as:

$$\tau = Y(q, \dot{q}, \dot{q}_{ref}, \ddot{q}_{ref})^T \hat{a} - K_D s \tag{30}$$

In the above equations, Γ, Λ, and K_D are all tuning parameters used to determine how quickly the adaptive parameters can change and how quickly position error is driven to zero. In general, the higher these values are driven, the faster the adaptive parameters change and the faster the position error is reduced. However, as one may expect, increasing these values too high can lead to instability.

Defining $f = M(q)\ddot{q}_{ref} + C(q, \dot{q})\dot{q}_{ref} + g(q) + K_d\dot{q} + K_{spring}q$, the regressor used for the continuum joint soft robot in this work is of the form:

$$Y(q, \dot{q}, \dot{q}_{ref}, \ddot{q}_{ref}) = \begin{bmatrix} \frac{\partial f}{\partial m} & \frac{\partial f}{\partial h^2} & \frac{\partial f}{\partial r^2} & \frac{\partial f}{\partial q} & \frac{\partial f}{\partial \dot{q}} \end{bmatrix}. \tag{31}$$

3) Model Reference Predictive Adaptive Control: MRPAC combines the strengths of both MPC and MRAC to yield a model-based optimal controller which can adapt its model online, but remains robust to unmodeled disturbances. As with MPC we begin with a model of the system, however this time we explicitly model the error in our model as a torque disturbance term:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w} + \tau_{disturbance}. \tag{32}$$

If the error in our model is simply due to incorrect estimates of the manipulator parameters, then we should be able to represent this disturbance exactly using the same regressor as MRAC, namely:

$$\tau_{disturbance} = -Y(q, \dot{q}, \dot{q}_{ref}, \ddot{q}_{ref})\hat{a}. \tag{33}$$

The negative sign is necessary because we adapt the parameters in $\hat{a}$ according to the MRAC adaptation law. MRAC's adaptation law is designed to estimate a torque which, when applied to the system, will "cancel out" the system's dynamics. In MRPAC we want to represent the system's dynamics themselves instead of the torque needed to cancel them out. These two quantities are opposite in sign, hence the negative sign shown here.

It is important to note here that in MRPAC we are using the regressor and adaptive parameters to represent our model error, while in MRAC they are used to represent the system dynamics in their entirety. We can not expect $\hat{a}$ therefore, to contain the same values for MRAC and MRPAC. In fact, if given a perfect model, $\hat{a}$ should theoretically remain zero for MRPAC.

Also, it is important to note that Γ and Λ are the only tuning parameters for the estimation of $\hat{a}$ in MRPAC. While in MRAC there must be an error term multiplied by K_D

in order to ensure that position error is decreased, in MPC the tracking error is decreased by virtue of the optimization which seeks to minimize it.

In order to make a fair comparison between MRAC and MRPAC we use the same regressor for both controllers.

III. Experiments

Adaptive control techniques are useful in the case where we do not know a priori a complete and accurate model of our system. After all, if we did have a complete and accurate model then we could predict perfectly the behavior of our system and use model-based control techniques to make it behave however we want. We will classify all modeling error into two categories: known unknowns and unknown unknowns. Known unknowns correspond to values or parameters in our model that we are accounting for, but whose values are uncertain or unknown. For example inertias, damping coefficients, and spring coefficients may be known unknowns. Unknown unknowns in our model correspond to phenomena which occur in the real system, but are not represented in our model. If we assume all spring and damping elements in our system are linear while they are in fact nonlinear, then we do not have the ability to represent the nonlinear effect of the spring and this nonlinear effect is an unknown unknown.

A. Simulation Experiments

In the simulation portion of the experiments, a simulation is created using the model outlined in Section II-B and this simulated system is controlled using three different controllers. The goal of each controller is to drive the system to follow a reference trajectory generated by a reference system. The three controllers implemented are MPC, MRAC, and the MRPAC algorithm detailed in Section II-C.3.

The reference system used for these experiments can be thought of as two uncoupled, critically-damped mass-spring-damper systems each modeled by the equation:

$$m\ddot{x} + b\dot{x} + k(x - r) = 0. \tag{34}$$

The masses (of mass m) are driven by the springs to the reference positions (r) and the damping coefficient (b) is always chosen such that the system is critically damped ($b = \sqrt{4mk}$). The rise time of the reference system can be altered by varying the spring constant (k). We choose a rise time such that the system has settled to steady state within about one second.

As mentioned in the adaptive control literature, model parameter estimation and adaptive control schemes require sufficient "excitation" in order to converge or to adapt. We provide this excitation by changing the reference positions (r) of our system every 2 seconds. Reference positions are drawn from a uniform distribution bounded above and below by $-\frac{\pi}{2\sqrt{2}}$ and $\frac{\pi}{2\sqrt{2}}$. These bounds are chosen so that the resulting total bend angle ($\phi = \sqrt{u^2 + v^2}$) is never greater than $\frac{\pi}{2}$.

1) Case 1: Perfect Regressor (known unknowns): The first experiment performed is designed to show the performance of all three controllers in the case where the regressor can fully describe the dynamics of the system. The hypothesis to be tested is that given a perfect regressor, both MRAC and MRPAC should be able to compensate for the system's dynamics perfectly and should drive the system to follow the reference trajectory exactly. Since MPC cannot adapt its model, we expect that increasing model error will lead to increasing tracking error.

To test this hypothesis we control the same system using the three controllers outlined in Section II-C (MPC, MRAC, and MRPAC) and provide each with the same regressor. Because MPC and MRPAC require a model, we introduce model error in order to see its effect on their performance. The method used for introducing model error is to make our estimates of h, m, K_{spring}, K_{damper} a scalar multiple of their simulated value. Because MRAC does not utilize a model apart from the regressor, it is invariant to model error. All adaptive parameters for MRAC and MRPAC are initialized at zero.

Each controller is run for simulated 5 minutes of "excitation" (new reference commands every 2 seconds) in order to allow the adaptive parameters to settle. After 5 minutes of "excitation" the performance of each controller is evaluated during one additional minute. The integrated position error during the evaluation minute is shown in Figure 3 as a function of the model error. As an example, the joint trajectories during the evaluation minute using a modeling error scalar of 1.5 is seen in Figure 4.

2) Case 2: Imperfect Regressor (unknown unknowns): The second experiment performed is designed to show the performance of all three controllers in the case where the regressor **cannot** fully describe the dynamics of the system. The hypothesis to be tested is that given an imperfect regressor, MRAC and MRPAC should not be able to compensate for

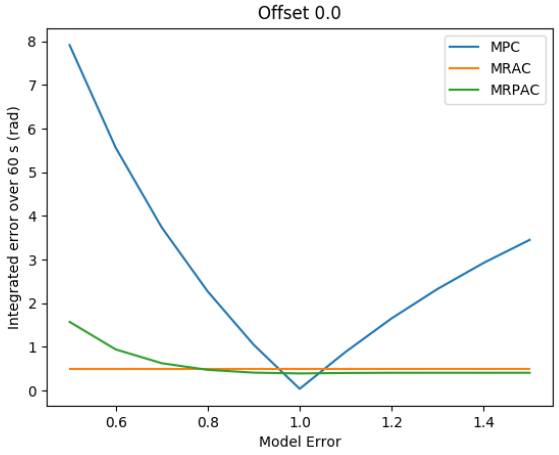

Fig. 3: Tracking error sensitivity to model error for all three controllers in simulation.

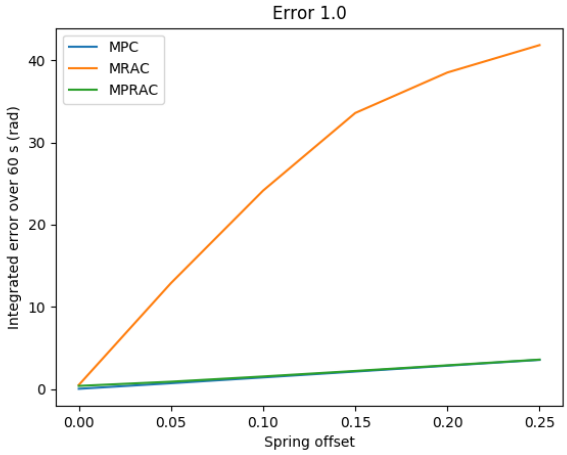

Fig. 5: Simulated tracking error sensitivity to unmodeled offset forces/torques (unknown unknowns) if the rest of the model is perfect.

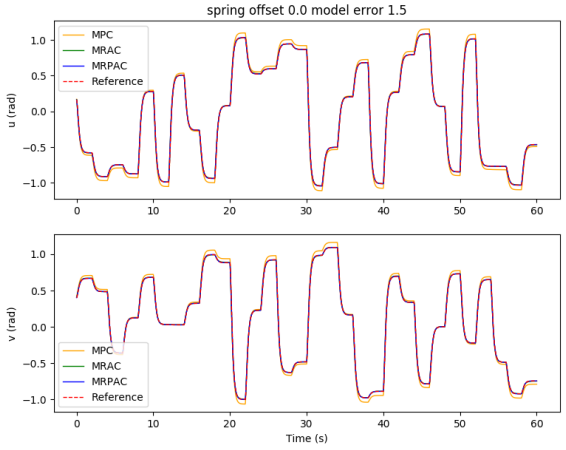

Fig. 4: Joint trajectory tracking using all three controllers in simulation. This is the case of errors in the model parameters used for MPC and MRPAC. Note that the performance of MRAC and MRPAC is indistinguishable.

the system's dynamics perfectly and should therefore struggle to drive the system to follow the reference trajectory exactly. However, because MPC has been shown to be robust to modeling error, both MPC and MRPAC should be robust to the unmodeled forces.

To test this hypothesis, instead of simulating a system in which a spring force drives the joint towards the zero configuration, we simulate a system in which the spring force drives the joint towards a nonzero configuration. This is a phenomenon observed in the real robot hardware because of slight inconsistencies in the manufacture of the plastic bellows. This offset spring force can be thought of as a constant torque which is applied to the joint in one direction. Because the regressor does not contain any terms which correspond to a constant torque offset, this force cannot be represented by the regressor and therefore constitutes an "unknown unknown". While we *do* know about this constant offset and likely would include a constant term in the regressor, we anticipate that there will be forces which we do not know about or whose form is unknown to us. This simple experiment allows us to see the potential effects of these completely unmodeled forces.

In order to see the sensitivity of each controller to this unmodeled force which cannot be represented with the regressor, we vary the offset between $u = v = .05$ rad and $u = v = .25$ rad. We do this for each model error tested in the first experiment, yielding a surface of tracking error which is a function of both a scaled model error as well as an unmodeled constant torque.

Again, after 5 minutes of "excitation" the performance of each controller is evaluated during one additional minute. The integrated position error during the evaluation minute is shown in Figure 5 as a function of the model error. As an example, the joint trajectories during the evaluation minute using a spring offset of $u = v = .25$ are seen in Figure 6.

B. *Hardware Experiments*

In order to validate both simulations, we implement the same three controllers (MPC, MRAC, and MRPAC) on the soft continuum joint shown in Figure 1 and compare their performance.

The soft continuum joint used for this experiment is

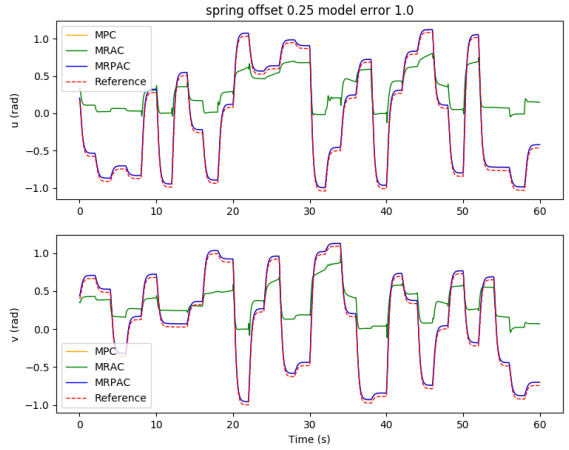

Fig. 6: Simulated joint trajectory tracking of all three controllers with a perfect model besides an unmodeled offset torque. Note that the performance of MPC and MRPAC is indistinguishable.

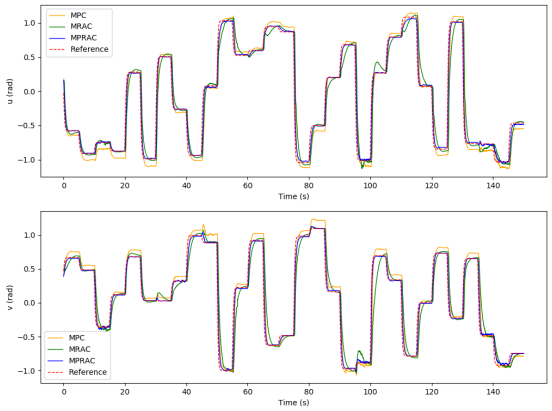

Fig. 7: Joint trajectory tracking of all three controllers in hardware.

actuated by four plastic bellows, each of which can be controlled independently. A pressure difference in each of the bellows causes a rotation about one or both of the joint's axes. The angle about each of these axes (denoted u and v in Figure 1) is the robot's position and what we attempt to control. We expect this hardware platform to illustrate the sensitivity of each controller to both known unknowns and unknown unknowns.

Both sources of error are present in hardware. Because no system identification was performed previously, the aforementioned model parameters such as h, m, K_{spring}, K_{damper} are not known perfectly. Additionally, we observe the effects of various offset forces and nonlinear behavior in the plastic bellows used to actuate the joint. For example, even with equal pressures in each of the four bellows, the continuum joint remains slightly bent, indicating some unmodeled force. This is simulated (see Section III-A.2) as a constant spring offset, but the actual source of this offset is unknown. Likewise, the continuum joint exhibits unknown nonlinear behavior near the extremes of its range of motion or in certain directions, where its stiffness or damping vary.

We track the orientation of a frame on top of the joint relative to a frame below the joint in order to estimate the state of the joint real-time. We reuse the same reference trajectory from the simulation with one minor change: the command changes every five seconds instead of every two. This was adjusted in an attempt to be conservative with experimental hardware and software while still validating the performance of each controller.

As in the simulation experiments, we excite the system with the same 150 commands used in simulation (12.5 minutes) before evaluating each of the controllers for the last 30 commands (2.5 minutes). The joint trajectories for this evaluation period are shown in Figure 7.

IV. Results

A. Simulation Experiments

1) Case 1: Perfect Regressor (known unknowns): The first experiment was designed to see the sensitivity of each controller to known unknowns, or model error where at least the form of the model is known. The results of this experiment can be seen in Figure 3. An example of the joint angle trajectories achieved by each controller is shown in Figure 4. As expected, MRAC is unaffected by this kind of model error because MRAC was initialized with all parameters equal to zero and adapted the parameters to their values based on the MRAC adaptation law. We see that given a correct form of the model, MRAC is able to find a very good model and track the reference trajectory with very little error. When MPC is given a perfect model, we see that it performs better than either MRAC or MRPAC, reducing tracking error to near zero over the entire evaluation period of 60 seconds. However we see that it is the most sensitive to model error, especially when inertial, damping, and spring effects are underestimated.

The data presented in Figure 3 seem to validate the hypothesis that MRAC and MRPAC can both compensate for model error, given a model with the perfect form. We see that MRPAC is able to perform almost identically to MRAC

in all cases except when inertial, damping, and spring effects are grossly underestimated. Upon further inspection of the data we found that for this case the adaptive parameters for MRPAC had not quite settled during the five minute excitation period and that given more time, the tracking performance of MRPAC again approached that of MRAC. This is an interesting and important note - that where MPC performs worst, MRPAC has the most tracking error to overcome, and therefore may take longer to converge its adaptive parameters to a steady state. This suggests that the transient responses of these controllers is an important topic of future research.

2) Case 2: Imperfect Regressor (unknown unknowns): The second experiment was designed to see the sensitivity of each controller to unknown unknowns, or model error where the form of the model is not completely known. The results of this experiment can be seen in Figure 5. An example of the joint angle trajectories achieved by each controller is shown in Figure 6. As can be seen from the figure, every controller's performance suffers because of this additional modeling error, however MRAC is by far the most sensitive. Note that the x axis of the plot denotes the value of both u and v, and the entire bend angle is equal to $\phi = \sqrt{u^2 + v^2}$. Keeping this in mind, with a spring offset of about 4° ($u = v = .05$) MRAC's tracking performance is worse than MPC with 50% error on estimates of masses, lengths and spring and damper coefficients. This represents a very significant decrease in performance due to a relatively small, but completely unmodeled, disturbance. This is the main motivation behind the development of MRPAC. MRPAC can be seen from this figure to inherit from MPC insensitivity to completely unmodeled disturbances or dynamics, and can be seen from Figure 3 to inherit from MRAC insensitivity to partially modeled disturbances or dynamics.

We can vary the magnitude of both scalar modeling error as well as the unmodeled spring offset in order to develop a surface of tracking error which is a function of both known unknowns and unknown unknowns. This surface can be seen in Figure 8. This is useful information because in reality we are likely to encounter both instead of just one. From the Figure we can see that MRPAC consistently has the lowest tracking error of the three controllers, except when MPC has a perfect model or when the model used for MRPAC grossly underestimates inertial, damping, and spring effects. As stated earlier, we have observed that the performance of MRPAC can be improved in the latter case by allowing it to adapt

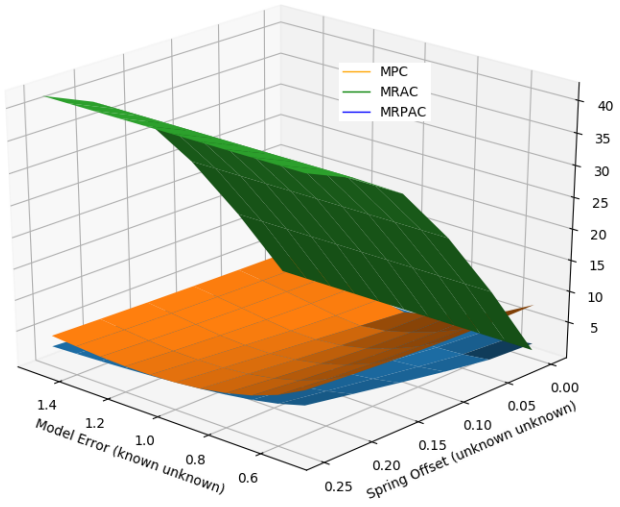

Fig. 8: Simulated joint trajectory tracking error as a function of both model parameter error (known unknowns) and a spring offset error (unknown unknowns)

for longer. However these experimental results do outline an important fact, which is that the transient responses of MRAC and MRPAC are not the same. The exact differences between them and the exact reasons remain for future work.

B. Hardware Experiments

The joint trajectories for the hardware experiments are shown in Figure 7. It is important to note that, unlike the simulation, we cannot separate the perfect regressor and imperfect regressor cases in the hardware. Because of the nature of the continuum joint, we expect some combination of both cases in the performance results.

Generally, we see from the results that MPC struggles to eliminate steady state error. This matches the behavior simulated in Figure 4 and is expected because MPC does not have the ability to compensate for unmodeled system dynamics which exist in the continuum joint. MRAC and MRPAC, on the other hand, do have the ability to compensate for unmodeled system dynamics. Consequently they both track the steady state reference trajectory much closer than MPC. This indicates that the hypothesis presented in Section III-A.1 is demonstratively true. MRAC and MRPAC certainly compensate for the system's dynamics and drive the system to follow the reference trajectory. They are incapable of following the reference trajectory *exactly* however, as is simulated in Figure 4, where both trajectories deviate very little from the reference. This is because of the impossibility of separating the test cases in hardware. MRAC and MRPAC

both compensate for unmodeled system dynamics (known unknowns) but there are still modeling errors (unknown unknowns) which cause these deviations.

The simulated effect of unknown unknowns is shown in Figure 5. Tracking error increases for all control methods as the magnitude of these modeling errors increase, but they increase dramatically for MRAC, hence its poor performance exhibited in Figure 6. This same pattern emerges in hardware experiments. There are several instances during the evaluation period where unknown forces cause deviation from the reference trajectory. For examples of this, see the upper plot (u) of Figure 7 at 65, 100, and 135 seconds and the bottom plot (v) at 30, 45, and 95 seconds. All controllers are negatively affected, but MPC and MRPAC are more robust than MRAC. In other words, when encountering such disturbances, MRAC is forced to artificially adapt dynamic parameters in an attempt to eliminate the error. In contrast, MPC and MRPAC are better able to filter disturbances because they re-solve the trajectory optimization over the whole time horizon, not just a single time step. These results indicate that the hypothesis outlined in Section III-A.2 is true as well. MRAC and MRPAC do not track the reference trajectory perfectly because of the unknown disturbances but MPC and MRPAC are more robust to them.

V. Conclusions and Future Work

In this paper we have presented a novel dynamic modeling approach for one joint of a continuum joint robot. We have shown that while not linear in the same parameters as rigid robots, joint accelerations using this model can be shown to be linear in other parameters. This linearity in model parameters can be exploited for system identification, or as we show later in the paper, for adaptive control. Future work in the area of continuum joint dynamic modeling may include system identification on hardware, as well as verification that the proposed model accurately describes the joint's dynamics. While the presented model is only valid for one joint, another straightforward extension to this work would be to derive the dynamic models using similar ideas and assumptions (constant curvature assumptions, u and v parameterization, and a symbolic math library) in order to derive a dynamic model for a robot with many joints and links.

In this paper we have also shown that MPC is an effective control strategy for controlling continuum joint soft robots with low-fidelity models. But the practical challenges of controlling soft robots are numerous. Medium to high fidelity models (such as the one presented in this paper) are promising, but are also time and labor intensive and may not even improve performance. Even equipped with a perfect model, determining soft robot model parameters accurately is a formidable task. As such, our presented control strategy, MRPAC, contributes a novel approach to overcome these challenges by adapting the dynamic model while still leveraging the benefits of MPC.

All told, MRPAC inherits two invaluable traits: the adaptive capabilities of MRAC and the robustness of MPC. As a result, MRPAC outperforms both MPC and MRAC on a soft continuum joint, where both known unknowns (such as unknown spring and damper coefficients) and unknown unknowns (such as unmodeled external forces or offsets) exist. MRPAC successfully compensates for modeling errors to eliminate steady state error while also demonstrating robustness to modeling disturbances.

Future research into MRPAC should include further investigation into how to identify a minimal regressor which still accurately represents a system's dynamics. Although not discussed in this work, the time taken by MRAC and MRPAC to converge to steady-state adaptive parameters was notably different for MRPAC it depended heavily on the model used to start with. The exact differences between the transient response of each control method as well as investigation into the reasons for these differences is left to future work.

Although the problems of accurate soft robot modeling and control remain interesting and unsolved problems, we believe that the dynamic model and adaptive control methods presented in this work represent an important contribution to the field.

References

[1] F. Renda, M. Cianchetti, M. Giorelli, A. Arienti, and C. Laschi, "A 3D steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm," *Bioinspiration and Biomimetics*, vol. 7, no. 2, 2012.

[2] T. G. Thuruthel, E. Falotico, M. Cianchetti, F. Renda, and C. Laschi, "Learning global inverse statics solution for a redundant soft robot," *ICINCO 2016 - 13th International Conference on Informatics in Control, Automation and Robotics, Doctoral Consortium*, vol. 2, pp. 303–310, 2016.

[3] Rongjie Kang, A. Kazakidi, E. Guglielmino, D. T. Branson, D. P. Tsakiris, J. A. Ekaterinaris, and D. G. Caldwell, "Dynamic model of a hyper-redundant, octopus-like manipulator for underwater applications," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4054–4059, 2011.

[4] W. Khalil, S. Member, and G. Gallot, "Dynamic Modeling and Simulation of a 3-D Serial Eel-Like Robot," vol. 37, no. 6, pp. 1259–1268, 2007.

[5] E. Tatlicioglu, I. D. Walker, and D. M. Dawson, "New dynamic models for planar extensible continuum robot manipulators," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1485–1490, 2007.

[6] I. S. Godage, D. T. Branson, E. Guglielmino, G. A. Medrano-Cerda, and D. G. Caldwell, "Shape function-based kinematics and dynamics for variable length continuum robotic arms," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 452–457, 2011.

[7] T. Zheng, D. T. Branson, R. Kang, M. Cianchetti, E. Guglielmino, M. Follador, G. A. Medrano-Cerda, I. S. Godage, and D. G. Caldwell, "Dynamic continuum arm model for use with underwater robotic manipulators inspired by Octopus vulgaris," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5289–5294, 2012.

[8] N. Giri and I. D. Walker, "Three module lumped element model of a continuum arm section," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4060–4065, 2011.

[9] I. D. Walker, "Continuous Backbone "Continuum" Robot Manipulators," *ISRN Robotics*, vol. 2013, pp. 1–19, 2013.

[10] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration & Biomimetics*, vol. 12, no. 6, p. 066003, 2017.

[11] H. Mochiyama and T. Suzuki, "Dynamical modelling of a hyper-flexible manipulator," in *Proceedings of the 41st SICE Annual Conference. SICE 2002.*, vol. 3. IEEE, 2002, pp. 1505–1510.

[12] ——, "Kinematics and dynamics of a cable-like hyper-flexible manipulator," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 3. IEEE, 2003, pp. 3672–3677.

[13] V. Falkenhahn, T. Mahl, A. Hildebrandt, R. Neumann, and O. Sawodny, "Dynamic modeling of constant curvature continuum robots using the euler-lagrange formalism," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2428–2433.

[14] ——, "Dynamic modeling of bellows-actuated continuum robots using the euler–lagrange formalism," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1483–1496, 2015.

[15] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[16] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A resilient, untethered soft robot," *Soft Robotics*, vol. 1, no. 3, pp. 213–223, 2014.

[17] H. Zhang, R. Cao, S. Zilberstein, F. Wu, and X. Chen, "Toward effective soft robot control via reinforcement learning," in *International Conference on Intelligent Robotics and Applications*. Springer, 2017, pp. 173–184.

[18] C. M. Best, M. T. Gillespie, P. Hyatt, L. Rupert, V. Sherrod, and M. D. Killpack, "A new soft robot control method: Using model predictive control for a pneumatically actuated humanoid," *IEEE Robotics & Automation Magazine*, vol. 23, no. 3, pp. 75–84, 2016.

[19] P. Hyatt, D. Wingate, and M. D. Killpack, "Model-Based Control of Soft Actuators Using Learned Non-linear Discrete-Time Models," *Frontiers in Robotics and AI*, vol. 6, p. 22, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2019.00022

[20] P. Hyatt and M. D. Killpack, "Real-Time Nonlinear Model Predictive Control of Robots Using a Graphics Processing Unit," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1468–1475, 2020.

[21] J.-S. Kim, "Recent advances in adaptive mpc," in *ICCAS 2010*. IEEE, 2010, pp. 218–222.

[22] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, "Adaptive mpc for iterative tasks," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6322–6327.

[23] J. J. E. Slotine and W. Li, "on the Adaptive Control of Robot Manipulators." *International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, 1987.

[24] J. S. Terry, J. Whitaker, R. W. Beard, and M. D. Killpack, "Adaptive control of large-scale soft robot manipulators with unknown payloads," in *ASME 2019 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

[25] R. L. Allen, T. and, T. Duggan, H. G., and K. Albert, "Closed-Form Non-Singular Constant-Curvature Continuum Manipulator Kinematics," *RoboSoft*, 2020.

[26] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[27] P. Hyatt, C. S. Williams, and M. D. Killpack, "Parameterized and gpu-parallelized real-time model predictive control for high degree of freedom robots," *arXiv preprint arXiv:2001.04931*, 2020.

[28] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Robust and adaptive control*. Springer, 2013, pp. 317–353.

[29] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *ArXiv e-prints*, Nov. 2017.