Tractable and Intuitive Dynamic Model for Soft Robots via the Recursive Newton-Euler Algorithm

Spencer W. Jensen[1], Curtis C. Johnson[1], Alexa M. Lindberg[1], Marc D. Killpack[1]

***Abstract*— Because of the complex nature of soft robots, formulating dynamic models that are simple, efficient, and sufficiently accurate for simulation or control is a difficult task. This paper introduces an algorithm based on a recursive Newton-Euler (RNE) approach that enables an accurate and tractable lumped parameter dynamic model. This model scales linearly in computational complexity with the number of discrete segments. We validate this model by comparing it to actual hardware data from a three-joint continuum soft robot (with six degrees of freedom represented in a constant curvature kinematic model). The results show that this RNE-based model can be computed faster than real-time. We also show that with minimal system identification, a simulation performed using the dynamic model matches the real robot data with a median error of 3.15 degrees.**

I. Introduction

Rigid-body equations of motion derived for rigid robots are well defined because of rigid-body assumptions that lead to simple lumped-parameter equations. Additionally, most rigid-body robot joints are modeled as having only one degree of freedom. These models are also relatively easy to evaluate when formulated as second-order ordinary differential equations. Soft-body dynamics on the other hand require understanding how the distribution of differential masses affects the overall movement and rotational acceleration of the body. To further complicate things, a continuum joint is often represented using two to three kinematic degrees of freedom. This leads to large and complex partial differential equations that can be difficult or computationally expensive to solve.

This paper presents a method of applying traditional rigid-body techniques to continuum or "soft" segments. We apply the derivation of the recursive Newton-Euler method for rigid-body forward and inverse dynamics, found in [1], to a robot with soft continuum joints that are pneumatically actuated and have rigid links. This model is not intended to replace more accurate and complex methods, but represents a way to formulate the dynamics that results in an intuitive, tractable, and sufficiently accurate method for simulation and control of serial hybrid (soft plus rigid-body) robot chains.

We expect this paper to provide the following contributions to the soft robot community:

1) extend the continuum joint modeling from [2] to a serial chain of continuum joints and rigid links
2) derive a sufficiently accurate yet simple lumped parameter dynamic model that remains tractable for real-time simulation
3) combine dynamic derivations and approaches from rigid and soft-body disciplines into a single straightforward method and presentation

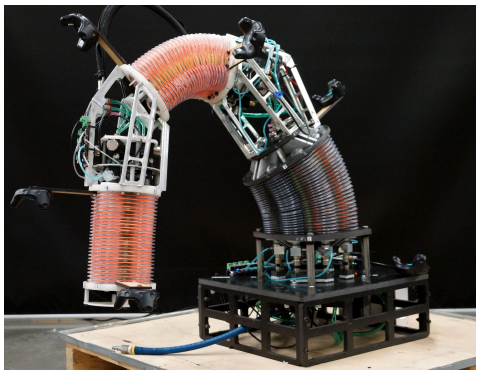

Fig. 1. The pneumatically driven soft robot arm used in this paper. Each orange segment is a continuum joint which is controlled by four independently actuated pressure chambers. The joints are interconnected with rigid aluminum links where the pneumatic valve assembly is mounted. At the end of each joint is a black HTC Vive Tracker used for ground truth joint angle estimation.

II. Related Work

Modeling a robot (whether for control or simulation) has two main components: kinematics and dynamics. A common assumption used to derive the kinematics of soft robots is to model the continuum joints as constant-curvature (CC) segments [3], [4]. This assumption is usually valid in the absence of large masses or external forces. There are many parameterizations of the kinematic space in the literature. Our paper uses the singularity free representation as defined in [5] which is equivalent to the form described in [6]. Once a valid parameterization to represent the position of the CC segment is found the velocities and accelerations with respect to each joint become a straightforward problem of taking derivatives.

Because of the wide variation in soft robot designs, the formulations for generating dynamic equations for soft robots also varies widely. In [7], the authors claim that the main concern while modeling soft robots is the trade-off between computational expense, model complexity, and accuracy. Several different methods have been introduced that explore these trade-offs. Newton-Euler (see [8]–[14]) and Euler-Lagrange (see [2], [15]–[17]) methods are the most traditionally implemented algorithms and depending on the implementation can vary widely in their computational ex-

[1]Robotics and Dynamics Lab, *Brigham Young University*, Provo, USA marc_killpack@byu.edu

This project was funded with funds provided by NSF EFRI and NRI

pense, complexity and accuracy. Because of the difficulty of modeling every single aspect of a soft robot accurately (hysteresis, friction, nonlinear material properties, etc.), several machine learning algorithms have also emerged to improve modeling soft robot motion [18]–[20]. Another widely used set of methods are finite-element methods which generally trade-off computational expense and model complexity for accuracy, although some researchers have proposed methods to reduce the computational burden [21]–[24].

This conference paper is a natural extension of the modeling work in [2] where the dynamic model of a single joint was analytically derived with the Euler-Lagrange method. This approach used the Sympy Python library (a symbolic mathematical manipulation tool) to analytically solve for the equations of motion and to then generate efficient code in the C language [25]. Unfortunately, this approach to generating entire equations of motion analytically became intractable to generate on a standard desktop computer as more continuum joints were added. This paper extends some of the main modeling ideas of [2] to a recursive Newton-Euler algorithm to efficiently compute the dynamics for a full hybrid multi-body robot arm with three continuum joints using six modeled kinematic degrees of freedom.

In this paper, we specifically use the recursive Newton-Euler (RNE) method as first introduced for rigid-body robots by [26]. The authors in [27] developed an RNE method for rigid robots that had elasticity in their joints, but did not include dynamics of continuum joints. In [28] they applied continuum body dynamics to rigid hyper-redundant robots and derived the momentum-balance equations that are the core of the Newton-Euler algorithm. In [10] and [9] the authors approximate a soft-body as a hyper-redundant rigid robot and use RNE to solve for the dynamics. However these methods require sufficient discretization to get accurate results. Khalil et al. showed an RNE derivation for soft and rigid-body systems, but the derivation includes the need to do three recursions because they assume a floating base with unknown acceleration [11], whereas our method only requires the traditional two recursions. Renda et al. have shown RNE methods combined with Cosserat/Kirchoff rod theory to model soft robots such as in [12]. However the joints we are modeling violate a primary assumption of Cosserat rods in that they would not be classified as slender rods because the joints have approximately the same diameter as length (see Figure 1). This is a necessary feature of the type of soft robots in which we are interested where the larger scale (in terms of diameter of the joints) allows for higher payloads and force output.

III. Dynamic Formulation

This section outlines the formulation of the dynamic equations for a hybrid (soft plus rigid segments) robot arm. First, the Jacobian and relevant kinematic parameters are detailed. Second, we show the equations for the forward pass of the recursive Newton-Euler algorithm. Third, we derive the Newton-Euler equations for a soft-body joint or rigid-body link that can be solved starting with the last link and working back to the base. Lastly, a special feature of the recursive Newton-Euler algorithm is that by running the algorithm several times with different conditions the standard form of (M, $C\dot{q}$, G) can be found. This form is desirable because it can be used for direct dynamics as well as in many control applications.

A. Kinematics

This derivation conforms to the frame convention described in [1]. The frames are placed in between the rigid and continuum segments such that the i^{th} frame is placed at the distal end of the i^{th} segment as shown in Figure 2. The authors, in [5], derive the kinematics for a single continuum joint. In the paper, they define u and v as the rotation of the distal end of the joint about the x and y axes respectively. In this paper, we use the same notation. The following variables which are defined in [5] are repeated here for clarity:

$$\begin{aligned} \phi &= \sqrt{u^2 + v^2} & \sigma &= cos(\phi) - 1 \\ \tilde{u} &= \frac{u}{\phi} & \tilde{v} &= \frac{v}{\phi} \end{aligned} \tag{1}$$

Given some minor discrepancies in the equations from [5], we explicitly redefine the relevant equations in this paper. The 3x3 rotation matrix R_i^{i-1}, denoted as the rotation from frame $i-1$ to frame i, for the continuum joints is calculated through

$$R_i^{i-1} = \begin{bmatrix} \sigma\tilde{v}^2 + 1 & -\sigma\tilde{u}\tilde{v} & \tilde{v}S_\phi \\ -\sigma\tilde{u}\tilde{v} & \sigma\tilde{u}^2 + 1 & -\tilde{u}S_\phi \\ -\tilde{v}S_\phi & \tilde{u}S_\phi & C_\phi \end{bmatrix} \tag{2}$$

where S_ϕ and C_ϕ are the sine and cosine functions of ϕ. It should be noted that the forward pass implements R_{i-1}^i which is the transpose of (2). The 3x1 vector from the base of the joint to the center of rotation, ρ, is calculated as

$$\rho = \frac{h}{\phi^2} \begin{bmatrix} v \\ -u \\ 0 \end{bmatrix} \tag{3}$$

where h is defined as the height or total length of the joint. For vectors, we use the following notation: superscripts refer to the frame that the vector is expressed in and the subscript represents the measured quantity. Additionally, the position vector from frame $i-1$, which is located at the center of the

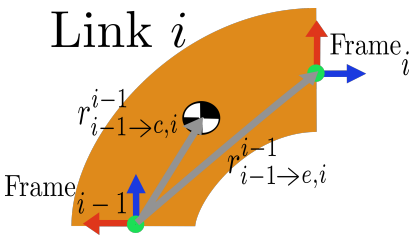

Fig. 2. Diagram of a continuum joint i to show the frame placement and position vectors from the base of the joint to the center of mass and distal end of the joint.

base of joint i, to the distal end of the same joint, denoted with a subscript e, expressed in the $i-1$ frame is defined as

$$r_{i-1\to e,i}^{i-1} = \begin{bmatrix} -\rho_x \sigma \\ -\rho_y \sigma \\ ||\rho|| S_\phi \end{bmatrix} \tag{4}$$

where the notation ρ_x and ρ_y is defined as the x and y components of ρ respectively. This vector is graphically defined in Figure 2. The previous equations can be found in [5].

Furthermore, the spatial Jacobian is defined as

$$J^s = \begin{bmatrix} 0 & \sigma h/\phi^2 \\ -\sigma h/\phi^2 & 0 \\ h\tilde{u}(\phi - S_\phi)/\phi^2 & h\tilde{v}(\phi - S_\phi)/\phi^2 \\ (\phi u^2 + S_\phi v^2)/\phi^3 & \tilde{u}\tilde{v}(\phi - S_\phi)/\phi \\ \tilde{u}\tilde{v}(\phi - S_\phi)/\phi & (\phi v^2 + S_\phi u^2)/\phi^3 \\ \sigma\tilde{v}/\phi & -\sigma\tilde{u}/\phi \end{bmatrix} \tag{5}$$

The derivation of which can be found in [5] or more generally in [29]. The spatial Jacobian is used because it can be calculated from simpler mathematical operations. For our purposes, we get the more common geometric Jacobian by using a shifting operation (sometimes called the adjoint in robotics literature, see [29]):

$$J = \begin{bmatrix} \mathbf{1} & -\left[r_{i-1\to e,i}^{i-1}\right]_x \\ \mathbf{0} & \mathbf{1} \end{bmatrix} J^s \tag{6}$$

where $\mathbf{1}$ and $\mathbf{0}$ represent a 3x3 identity and null matrix respectively.

By the definition of the geometric Jacobian, for a continuum joint, i, the linear and angular velocity, $\nu_{e,rel}^{i-1}$ and $\omega_{e,rel}^{i-1}$, as well as the linear and angular acceleration, $a_{e,rel}^{i-1}$ and $\alpha_{e,rel}^{i-1}$, of the distal end of the link relative to and in the base frame of the joint, $i-1$, is calculated by

$$\begin{aligned} \begin{bmatrix} \nu_{e,rel}^{i-1} \\ \omega_{e,rel}^{i-1} \end{bmatrix} &= J\dot{q} \\ \begin{bmatrix} a_{e,rel}^{i-1} \\ \alpha_{e,rel}^{i-1} \end{bmatrix} &= J\ddot{q} + \dot{J}\dot{q} \end{aligned} \tag{7}$$

where q and $\dot{q}$ are the vector of joint rotations and velocities of joint, i, respectively. The Jacobian derivative is found via the chain rule with

$$\dot{J} = \frac{\partial J}{\partial u}\dot{u} + \frac{\partial J}{\partial v}\dot{v} \tag{8}$$

The velocity and acceleration vectors of the center of mass of the joint can be found by replacing the vector to the distal end of the joint with the vector to the center of mass in both the transformation matrix, g, and the Jacobian, J, as defined in [29], and [5]. The position vector from frame $i-1$ to the center of mass, denoted as a subscript c, of link i relative to frame $i-1$, denoted as $r_{i-1\to c,i}^{i-1}$, is calculated as follows:

$$r_{i-1\to c,i}^{i-1} = \frac{h}{\phi^2}\begin{bmatrix} (\phi - S_\phi)\frac{v}{\phi} \\ (\phi - S_\phi)\frac{-u}{\phi} \\ (1 - C_\phi) \end{bmatrix} \tag{9}$$

which is taken from [2] and shown graphically in Figure 2.

The relative velocity and acceleration of the distal point is with respect to and described in the rotating frame at the base of the link. This means that for a rigid link, i, the relative velocity and acceleration vectors for both the distal end and center of mass are set to a 3x1 zero vector such that

$$\begin{bmatrix} \nu_{rel} \\ \omega_{rel} \end{bmatrix} = \mathbf{0} \qquad \begin{bmatrix} a_{rel} \\ \alpha_{rel} \end{bmatrix} = \mathbf{0} \tag{10}$$

B. Forward Recursion

The forward pass iteratively calculates the linear and angular accelerations of the joint. The following five equations are calculated for each joint or rigid link going from the base link/joint up to the n^{th} link/joint for an n-link/joint serial robot. The recursion is started by setting the velocities and accelerations at the base of the serial robot arm to a previously known velocity and acceleration, usually zero.

For the i^{th} link:

$$\omega_i^i = R_{i-1}^i \omega_{i-1}^{i-1} + R_{i-1}^i \omega_{i,rel}^{i-1} \tag{11}$$

$$\alpha_i^i = R_{i-1}^i \alpha_{i-1}^{i-1} + R_{i-1}^i a_{i,rel}^{i-1} + \omega_i^i \times R_{i-1}^i \omega_{i,rel}^{i-1} \tag{12}$$

$$\nu_{e,i}^i = R_{i-1}^i (\nu_{e,i-1}^{i-1} + \nu_{e,rel}^{i-1} + \omega_{i-1}^{i-1} \times r_{i-1\to e,i}^{i-1}) \tag{13}$$

$$\begin{aligned} a_{c,i}^i =& R_{i-1}^i (a_{e,i-1}^{i-1} + a_{c,rel}^{i-1} + \left(\alpha_{i-1}^{i-1} \times r_{i-1\to c,i}^{i-1}\right) \\ &+ \left[\omega_{i-1}^{i-1} \times \left(\omega_{i-1}^{i-1} \times r_{i-1\to c,i}^{i-1}\right)\right] + 2\omega_{i-1}^{i-1} \times \nu_{c,rel}^{i-1}) \end{aligned} \tag{14}$$

$$\begin{aligned} a_{e,i}^i =& R_{i-1}^i (a_{e,i-1}^{i-1} + a_{e,rel}^{i-1} + \left(\alpha_{i-1}^{i-1} \times r_{i-1\to e,i}^{i-1}\right) \\ &+ \left[\omega_{i-1}^{i-1} \times \left(\omega_{i-1}^{i-1} \times r_{i-1\to e,i}^{i-1}\right)\right] + 2\omega_{i-1}^{i-1} \times \nu_{e,rel}^{i-1}) \end{aligned} \tag{15}$$

where the variables are defined in section III-A. Equations 11-15 are taken from [30] for the velocity and acceleration of a point in a rotating frame.

C. Backward Recursion

Once the accelerations at the center of mass of each link are known, the forces and torques can be calculated backwards recursively from the n^{th} link down to the base of the 1^{st} link. The equations to find the forces and torques at the beginning of the i^{th} link are

$$f_i^i = R_{i+1}^i f_{i+1}^{i+1} - m_i R_0^i g^0 + m_i a_{c,i}^i \tag{16}$$

$$\begin{aligned} \tau_i^i =& R_{i+1}^i \tau_{i+1}^{i+1} - f_i^i \times \left(R_{i-1}^i r_{i-1\to c,i}^{i-1}\right) \\ &+ R_{i+1}^i f_{i+1}^{i+1} \times \left[R_{i-1}^i \left(r_{i-1\to c,i}^{i-1} - r_{i-1\to e,i}^{i-1}\right)\right] + \dot{H} \end{aligned} \tag{17}$$

where f, τ, and g are 3x1 Cartesian force, torque, and gravitational acceleration vectors. m_i is the mass of the i^{th} segment and $\dot{H}$ is the time derivative of angular momentum. Equivalent equations can be found in [30] or [1] and are based on Newton-Euler methods. It should be noted that there is an important assumption in these equations. These equations govern the average motion of a system or body of particles [30]. More accurate results could be found by including the interaction forces between all particles in the joint however this would significantly complicate the

model. We assume that the average motion of the particles enables sufficiently accurate simulation. We propose that rapid evaluation of these equations makes this a justifiable assumption.

For the calculation of $\dot{H}$, we assume that at each time step the continuum joint can be approximated as a rigid link. This is calculated simply as

$$\dot{H} = I^i_{i,c}\alpha^i_i + \omega^i_i \times \left(I^i_{i,c}\omega^i_i\right) \tag{18}$$

which can be found in [30]. The 3x3 inertia tensor of the i^{th} link about the center of mass and in the i^{th} frame, $I^i_{i,c}$, is updated at each time step and is defined as

$$I^i_{i,c} = \int_0^h (R^i_{i-1}R^{i-1}_{disk}I^{disk}_{disk,c}R^{i-1\,T}_{disk}R^{i\,T}_{i-1} - \mu\left[r^i_{i,c\to disk,c}\right]_x\left[r^i_{i,c\to disk,c}\right]_x)\,dl \tag{19}$$

which uses the infinitely thin disk assumption and uniform linear density, $\mu = m/h$, as in [2], with the parallel axis theorem to move the moments of inertia of each disk to the center of the joint. Additionally, $[\,]_x$ represents the skew symmetric operator which forms a skew symmetric matrix from a 3x1 vector. The integration is performed from zero to h over l which is defined as the length along the central arc of the joint. The inertia tensor of an infinitely thin disk about the center of mass of the disk, $I^{disk}_{disk,c}$, is defined in [2] as

$$I^{disk}_{disk,c} = \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \mu(D/2)^2 \tag{20}$$

where D is the diameter of the disk. The vector of positions from the center of the joint to the center of each disk, $r^i_{i,c\to disk}$, is calculated by

$$r^i_{i,c\to disk} = R^i_{i-1}\left(r^{i-1}_{i-1\to disk,c} - r^{i-1}_{i-1\to i,c}\right) \tag{21}$$

and the position vector from the base of the joint to the center of mass of each disk , $r^{i-1}_{i-1\to disk,c}$, is defined as

$$r^{i-1}_{i-1\to disk,c} = \begin{bmatrix} -\rho_x\sigma_l \\ -\rho_y\sigma_l \\ ||\rho||S_{\phi_l} \end{bmatrix} \tag{22}$$

where ϕ_l is defined in [2] as $l/||\rho||$ and σ_l is defined the same as σ in (1) except with ϕ_l instead of ϕ. In practice, the algorithm is written in C code for rapid execution, however (6), (8), and (19) are computed symbolically in Sympy and then written to C code [25].

D. Lagrangian Form

The inverse dynamic problem, useful in feed-forward control applications, of getting input torques given joint angles, velocities, and accelerations is as simple as doing one pass forward and backward. In order to do direct dynamics, for simulation, and for some model-based control methods we would like to get the equations of motion in a form like the following:

$$\ddot{q} = M^{-1}(-C\dot{q} - G + K_{prs}p - K_{spring}q - K_d\dot{q}) \tag{23}$$

where q is a vector of the joint angles (or generalized coordinates) and $\dot{q}$ and $\ddot{q}$ are its time derivatives. Following the convention in [31], M is the mass or inertial matrix, C is the Coriolis matrix, G is the vector of torques due to gravitational loads and all of these arrays can be calculated via the recursive Newton-Euler algorithm explained above. As defined in [2], K_{prs}, K_{spring}, and K_d are matrices describing the torque-to-pressure mapping, the parasitic spring coefficients, and the damping parameters of a soft-body joint respectively. These matrices are a function of the construction of the soft robot and are found via system identification.

M, $C\dot{q}$, and G can be calculated by setting different initial conditions and running the recursive Newton-Euler algorithm multiple times. To calculate:

- G, we set $\dot{q} = \ddot{q} = \mathbf{0}$.
- $C\dot{q}$, we set $\ddot{q} = g = \mathbf{0}$.
- for the i^{th} column of M, we set $\ddot{q}_i = 1$, all other $\ddot{q}_{-i} = 0$, and $\dot{q} = g = \mathbf{0}$.

The resulting torques at each of the defined frames calculated by the recursive Newton-Euler algorithm with the conditions specified above correspond to the $n \times 1$ vectors of G, $C\dot{q}$, and the i^{th} column of M where n refers to the number of torque inputs.

IV. Experimental Setup

A. Hardware Description

We validate our dynamic model formulation using data from the hybrid continuum robotic arm shown in Figure 1. The robot is composed of three pneumatically-actuated continuum joints connected serially by rigid links. Each joint uses four blow-molded pressure chambers which surround an inextensible steel cable as is shown in Figure 3. Pressure differentials between opposing chambers create a net torque which bends the joint about its x and y axes (as described in Figure 3) with constant curvature.

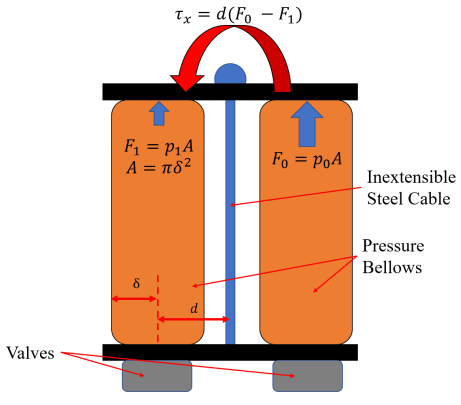

Fig. 3. Diagram of pneumatic joint and blow-molded pressure chambers. This joint is shown sliced along the y-z plane, where z points upwards, y points to the right, and x points out of the page.

Figure 3 shows the pressure-to-torque mapping for a single joint of the three continuum joint soft robot. The straight blue arrows represent the forces being applied by the pressure chambers and the curved red arrow represents the equivalent torque about the x axis. Because the equivalent forces would be acting along the inextensible steel cable these forces are neglected. Applying this derivation to calculate the torque about the y axis and putting the result in matrix form gives the following equations for the torque of a single joint:

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} = \begin{bmatrix} dA & -dA & 0 & 0 \\ 0 & 0 & dA & -dA \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = K_{prs} p \tag{24}$$

where the meaning of the variables is shown in Figure 3. d is the distance from the center of the joint to the center of the pressure chamber, and p is the pressure inside the pressure chamber. A is the cross-sectional area of a single pressure chamber where $A = \pi\delta^2$ and δ is the radius of the pressure chamber.

B. Data Collection

We collected data on the pneumatic robot arm shown in Figure 1 by sending 5 randomized 2 second pressure step commands between 0 and 150 KPa. A representative input trajectory can be seen in the top graph in Figure 4, which shows the actual pressures for joint 1. HTC Vive Trackers, as seen in Figure 1, provide the resulting rotation matrix from the base to the distal end of each continuum joint. The elements of this matrix are used to solve for the joint angles, u and v using equation 2. We refer to this dataset as the training dataset. We then used Optuna [32], a parameter optimization software package, to estimate joint spring (K_{spring}) and damping coefficients (K_d) as seen in (23).

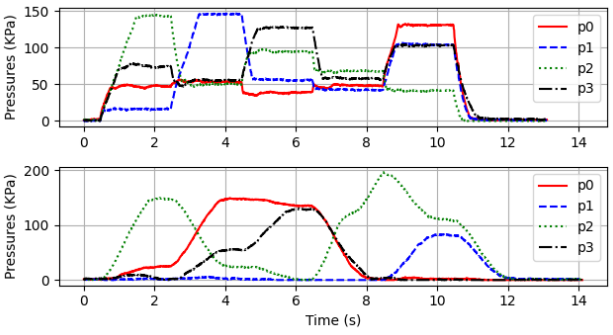

Fig. 4. Graphs of the pressure trajectory for the first joint. Top graph shows the step trajectory which is used to get the training dataset. Bottom graph shows the smoothed input trajectory for the test dataset. Joint 2 and 3, not shown, have similar trajectories.

In order to test the ability of the presented modeling approach to generalize beyond step commands, we simulated the model using a set of test data (i.e. data which was not used for system identification). This data consisted of smooth pressure trajectories over random areas of the robot's workspace.

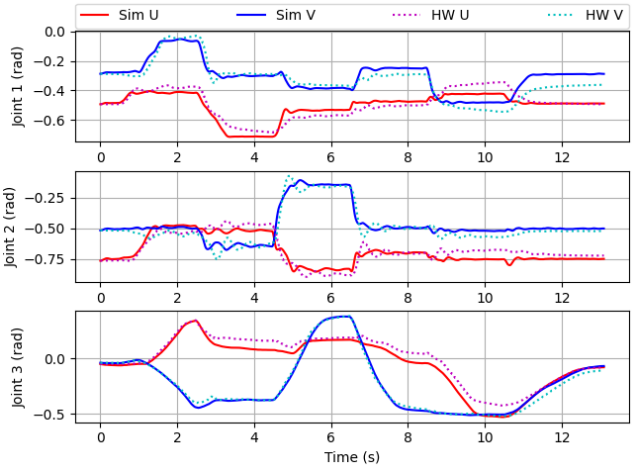

Fig. 5. Graph of the joint angles of the three pneumatically actuated joints with six kinematic degrees of freedom in our model. The predicted joint angles from the simulation are plotted in solid lines and the results from the hardware training data are plotted as dotted lines. The hardware results were used to optimize the joint stiffness (K_{spring}) and damping (K_d) matrices.

TABLE I
ERROR STATISTICS BETWEEN SIMULATION AND HARDWARE ON TRAINING DATA

Joint Angles	Mean (rad)	Median (rad)	Max (rad)
u_1	0.035	0.035	0.102
v_1	0.041	0.036	0.106
u_2	0.043	0.040	0.127
v_2	0.036	0.025	0.137
u_3	0.047	0.044	0.120
v_3	0.019	0.012	0.071
$Total$	0.037	0.031	0.137

V. RESULTS

The previously outlined algorithm in section III was written and compiled in the C/C++ language to form the equations of motion and wrapped in python code in order to perform numerical integration using the *scipy.integrate.solve_ivp* function [33] with the *BDF* method [34]. The simulation uses multiple cores on an AMD Ryzen 9 5900 12-Core Processor. The actual pressure trajectories from the training and the test datasets were applied to the simulation. Starting with the training dataset, Figure 5 shows the simulation joint angle trajectories (solid lines) over time, u (bending about the x-axis) in red and v (bending about the y-axis) in blue, plotted against the hardware training data shown with dotted lines. This dataset was used for optimizing joint spring (K_{spring}) and damping coefficients (K_d) as mentioned in Section IV-B. Table I shows the mean, median and maximum absolute error between the simulation and the hardware for the training dataset. As shown in Table I, the total error between the simulation and the hardware has a median of 0.031 radians or approximately 1.8 degrees. The simulation runnning on the

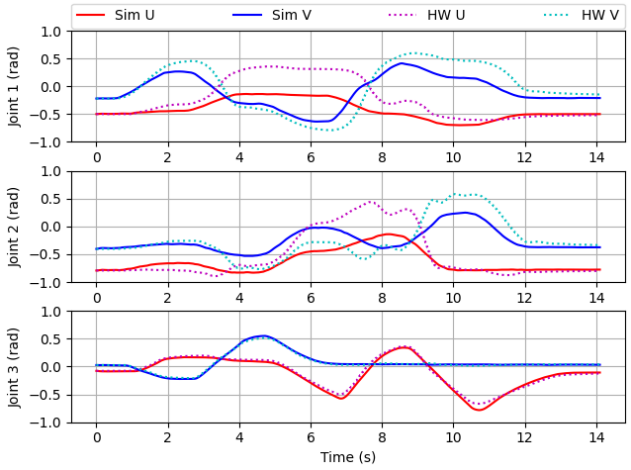

Fig. 6. Graph of the joint angles from the three actuated joints. The predicted joint angles from the simulation are plotted in solid lines and the results from the hardware data are plotted as dotted lines. As can be seen in the graph, the simulation matches the trends of the physical hardware. The simulation's constant stiffness and damping matrix have been optimized using data from Figure 5. However it should be noted that Figure 5 has a much smaller overall displacement than the above graph.

TABLE II

ERROR STATISTICS BETWEEN ENTIRE SIMULATED JOINT TRAJECTORY AND ACTUAL JOINT TRAJECTORY ON TEST DATA.

Joint Angles	Mean (rad)	Median (rad)	Max (rad)
u_1	0.207	0.121	0.587
v_1	0.165	0.131	0.425
u_2	0.181	0.105	0.663
v_2	0.166	0.107	0.451
u_3	0.033	0.026	0.114
v_3	0.012	0.012	0.041
$Total$	0.127	0.055	0.663

training dataset had a mean completion time of 9.77 seconds or 74.6 percent of real-time with a sample of 30 runs.

For the test dataset, Figure 6 shows the results of the simulation versus the hardware. Table II shows the average, median and maximum absolute error between the simulation and the physical robot arm for the test dataset. The total median absolute error was 0.055 radians or 3.15 degrees. The simulation had a mean completion time of 12.02 seconds or 85.2 percent of real-time with a sample of 30 runs.

VI. Discussion and Conclusion

Figures 5 and 6 indicate that the dynamic model presented in this paper captures overall trends reasonably well. This is supported by the statistics reported in Tables I and II where the worst case median absolute error for any single joint angle across all six joint angles for the test data was 0.131 radians (approximately 7.5 degrees on joint angle v_1).

On the other hand, it is clear that the model does not capture some higher order dynamic modes of the system and occasionally struggles to reach the correct steady-state values. This is most clearly seen in the Joint 1 and Joint 2 subplots in Figures 5 and 6 where the maximum absolute errors are significantly higher.

We hypothesize that this error is likely due to unmodeled dynamics in the terms most related to our actuator model. These terms include K_{prs}, K_{spring}, and K_d which are a part of (23). These three terms are not part of the recursive algorithm presented in this paper but are instead linear approximations of actuator dynamics that we expect could more accurately be represented with nonlinear or higher-order terms (including phenomenon like pressures waves in the large pneumatic chambers).

The effects of these terms are most noticeable in the dynamic response of Joint 1 and Joint 2. We hypothesize that the linear pressure-to-torque mapping term $K_{prs}p$ is actually a function of q and possibly $\dot{q}$ as well. That is, the joint dynamics are coupled with the pressure dynamics. To explain this dynamic coupling, consider a pressure chamber which compresses as q changes. This compresses the air inside of it briefly before the valves are able to vent the extra pressure. But as the arm continues to oscillate, the same chamber that was compressed stretches out causing a drop in pressure. The valves would subsequently open to maintain the commanded pressure, potentially causing the small oscillation in joint angle that is evident in the data. In addition, from prior data we expect that the linear spring and viscous damping terms ($K_{spring}q$ and $K_d\dot{q}$) are nonlinear–especially near joint limits as pressure chambers reach their maximum or minimum deformations.

This explanation is supported by the fact that in both input trajectory cases, the maximum absolute error for Joint 3 is less than 0.07 radians (or 4 degrees), showing that our proposed model captures the dynamics well. Joint 3 is the most distal joint on the arm, where we expect the proposed joint-pressure dynamic coupling to be least influential because this joint carries the least mass. This also means that the joint is less likely to reach joint limits where the nonlinearity in stiffness and damping are most pronounced. Because these actuator dynamics are not the main contribution of this paper, we leave their modeling and identification for future work.

In conclusion, in this paper, we have presented a recursive dynamic soft robot model that is tractable with faster than real-time execution. In addition, the model accurately represents the effect of distributed mass (e.g. rotational inertia) on large-scale soft continuum joints with rigid links. Minor errors (in terms of amplitude) in the model deserve additional attention in future work. However, the current results show that this model is scalable to complex soft robot manipulators and may be used in future applications for both simulation and model-based control given the level of accuracy that we have already demonstrated.

References

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics modelling, planning and Control*. Springer, 2009.

[2] 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. 132, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.558027

[3] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control Strategies for Soft Robotic Manipulators: A Survey," *Soft Robotics*, vol. 5, no. 2, pp. 149–163, Apr. 2018. [Online]. Available: https://www.liebertpub.com/doi/10.1089/soro.2017.0007

[4] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, Nov. 2010, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364910368147

[5] T. F. Allen, L. Rupert, T. R. Duggan, G. Hein, and K. Albert, "Closed-Form Non-Singular Constant-Curvature Continuum Manipulator Kinematics," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, New Haven, CT, USA, May 2020, pp. 410–416, tex.ids: allenClosedFormNonSingularConstantCurvature2020.

[6] C. D. Santina and D. Rus, "Control Oriented Modeling of Soft Robots: The Polynomial Curvature Case," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 290–298, Apr. 2020, conference Name: IEEE Robotics and Automation Letters.

[7] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum Robots for Medical Applications: A Survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, Dec. 2015, conference Name: IEEE Transactions on Robotics.

[8] G. Le Vey, "Optimal control theory and Newton–Euler formalism for Cosserat beam theory," *Comptes Rendus Mécanique*, vol. 334, no. 3, pp. 170–175, Mar. 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1631072106000076

[9] W. Khalil, G. Gallot, and F. Boyer, "Dynamic Modeling and Simulation of a 3-D Serial Eel-Like Robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1259–1268, Nov. 2007, conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[10] R. Kang, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamic modeling and control of an octopus inspired multiple continuum arm robot," *Computers & Mathematics with Applications*, vol. 64, no. 5, pp. 1004–1016, Sept. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0898122112002234

[11] W. Khalil, F. Boyer, and F. Morsli, "General Dynamic Algorithm for Floating Base Tree Structure Robots With Flexible Joints and Links," *Journal of Mechanisms and Robotics*, vol. 9, no. 3, Mar. 2017. [Online]. Available: https://doi.org/10.1115/1.4035798

[12] F. Renda and L. Seneviratne, "A Geometric and Unified Approach for Modeling Soft-Rigid Multi-Body Systems with Lumped and Distributed Degrees of Freedom," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1567–1574, iSSN: 2577-087X.

[13] J. Till, V. Aloi, and C. Rucker, "Real-time dynamics of soft and continuum robots based on Cosserat rod models," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 723–746, May 2019, zSCC: 0000025 Publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364919842269

[14] F. Boyer, V. Lebastard, F. Candelier, and F. Renda, "Dynamics of Continuum and Soft Robots: A Strain Parameterization Based Approach," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 847–863, June 2021, conference Name: IEEE Transactions on Robotics.

[15] V. Falkenhahn, T. Mahl, A. Hildebrandt, R. Neumann, and O. Sawodny, "Dynamic Modeling of Bellows-Actuated Continuum Robots Using the Euler–Lagrange Formalism," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1483–1496, Dec. 2015, conference Name: IEEE Transactions on Robotics.

[16] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamics for variable length multisection continuum arms," *The International Journal of Robotics Research*, vol. 35, no. 6, pp. 695–722, May 2016, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364915596450

[17] A. Ehsani-Seresht and S. Hashemi-Pour Moosavi, "Dynamic Modeling of the Cable-Driven Continuum Robots in Hybrid Position-Force Actuation Mode," *Journal of Mechanisms and Robotics*, vol. 12, no. 5, Mar. 2020. [Online]. Available: https://doi.org/10.1115/1.4046252

[18] 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," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2018, pp. 39–45.

[19] C. C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. D. Killpack, "Using First Principles for Deep Learning and Model-Based Control of Soft Robots," *Frontiers in Robotics and AI*, vol. 8, 2021, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.654398/full

[20] A. Tariverdi, V. K. Venkiteswaran, M. Richter, O. J. Elle, J. Tørresen, K. Mathiassen, S. Misra, and O. G. Martinsen, "A recurrent neural-network-based real-time dynamic model for soft continuum manipulators," *Frontiers in Robotics and AI*, vol. 8, p. 45, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2021.631303

[21] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3982–3987, iSSN: 1050-4729.

[22] E. Coevoet, A. Escande, and C. Duriez, "Optimization-Based Inverse Model of Soft Robots With Contact Handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, July 2017, conference Name: IEEE Robotics and Automation Letters.

[23] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, and C. Duriez, "Software toolkit for modeling, simulation, and control of soft robots," *Advanced Robotics*, vol. 31, no. 22, pp. 1208–1224, Nov. 2017, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01691864.2017.1395362. [Online]. Available: https://doi.org/10.1080/01691864.2017.1395362

[24] M. Pozzi, E. Miguel, R. Deimel, M. Malvezzi, B. Bickel, O. Brock, and D. Prattichizzo, "Efficient FEM-Based Simulation of Soft Robots Modeled as Kinematic Chains," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 4206–4213. [Online]. Available: https://ieeexplore.ieee.org/document/8461106/

[25] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "Sympy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017. [Online]. Available: https://doi.org/10.7717/peerj-cs.103

[26] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-Line Computational Scheme for Mechanical Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, no. 2, pp. 69–76, June 1980. [Online]. Available: https://doi.org/10.1115/1.3149599

[27] G. Buondonno and A. De Luca, "A recursive Newton-Euler algorithm for robots with elastic joints and its application to control," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2015, pp. 5526–5532.

[28] G. S. Chirikjian, "Hyper-redundant manipulator dynamics: a continuum approximation," *Advanced Robotics*, vol. 9, no. 3, pp. 217–243, Jan. 1994, publisher: Taylor & Francis _eprint: https://doi.org/10.1163/156855395X00175. [Online]. Available: https://doi.org/10.1163/156855395X00175

[29] R. M. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[30] J. H. Ginsberg, *Engineering dynamics*. Cambridge Univ. Press, 2008.

[31] P. Corke, *Robotics, vision and control fundamental algorithms in MATLAB*. Springer International Publishing, 2017.

[32] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019.

[33] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[34] G. D. Byrne and A. C. Hindmarsh, "A polyalgorithm for the numerical solution of ordinary differential equations," *ACM Trans. Math. Softw.*, vol. 1, no. 1, p. 71–96, Mar. 1975. [Online]. Available: https://doi.org/10.1145/355626.355636