Model-Predictive Control with Reference Input Tracking for Tensegrity Spine Robots

Andrew P. Sabelhaus, Huajing Zhao, Edward L. Zhu, Adrian K. Agogino, Alice M. Agogino

Abstract—Robots with flexible spines based on tensegrity structures have potential advantages over traditional designs with rigid torsos. However, these robots can be difficult to control due to their high-dimensional nonlinear dynamics. To overcome these issues, this work presents two controllers for tensegrity spine robots, using model-predictive control (MPC), and demonstrates the first closed-loop control of such structures. The first of the two controllers is formulated using only state tracking with smoothing constraints. The second controller, newly introduced in this work, tracks both state and input reference trajectories without smoothing. The reference input trajectory is calculated using a rigid-body reformation of the inverse kinematics of tensegrity structures, and introduces the first feasible solutions to the problem for certain tensegrity topologies. This second controller significantly reduces the number of parameters involved in designing the control system, making the task much easier. The controllers are simulated with 2D and 3D models of a particular tensegrity spine, designed for use as the backbone of a quadruped robot. These simulations illustrate the different benefits of the higher performance of the smoothing controller versus the lower tuning complexity of the more general input-tracking formulation. Both controllers show noise insensitivity and low tracking error, and can be used for different control goals. The reference input tracking controller is also simulated against an additional model of a similar robot, thereby demonstrating its generality.

Index Terms—Predictive control, robot control, robot kinematics, robot motion, soft robotics

I. INTRODUCTION

Quadruped (four-legged) robots that are designed with rigid torsos have limitations on the type of terrain on which they can walk [1], [2], [3]. Alternatively, when quadruped robots are constructed with spine-like flexible bodies that include actuation, significant control challenges are encountered. Few model-based closed-loop control approaches have been developed, and results instead used kinematics-only models [4], [5], model-free control using machine learning [6], [7], [8], or the replaying of open-loop inputs [9] among others.

The Laika project, named for the first dog in space, is an ongoing research effort to develop a quadruped robot with a flexible, actuated body that can be controlled to track specific movements in closed-loop [10]. Laika is designed to use a flexible, actuated spine as its body, developed as a form of the previously-investigated ULTRA Spine (Underactuated Lightweight Tensegrity Robot Spine) [11], [12]. This work presents two controllers for closed-loop trajectory tracking of Laika’s spine using model-predictive control (MPC), extending the benefits of the first controller [12] to a more general approach in the second.

A. Tensegrity Robots and Control

The spine robot studied in this work (Fig. 1) is a tensegrity, or "tension-integrity", structure. Tensegrity structures consist of rigid bodies suspended in a network of cables in tension such that no two bodies contact each other [13]. Such structures are inherently flexible, and many types of tensegrity robots have been designed that leverage this flexibility. These robots are able to adjust the lengths of their cables to roll [14], [15], crawl [16], [17], [18], swim [19], and climb [20], [21]. Tensegrity spine robots have been previously investigated [22], [18], but the ULTRA Spine and its recent adaptation for Laika are one of the first uses of a tensegrity spine on a quadruped robot [11], [8].
Control of tensegrity robots has proven challenging, no matter their shape or type of motion. Model-based closed-loop control has been mostly limited to low-dimensional structures [23], [24], [25], [13], [26]. More complex and high dimensional systems have been addressed with model-free methods [16], [22], [18], [27], [28], [29] or open-loop control [30], [31], [32], [20]. In order to use a tensegrity spine with Laika, a model-based closed-loop tracking controller was developed by the authors in [12] and is improved upon in this work.

**B. Model-Predictive Control Formulations for Tensegrity Spine Robots**

This work presents two different controllers for Laika’s spine, both based on model-predictive control. As motivated by previous robotics applications [33], [34], MPC was initially chosen to introduce smoothing constraints upon an optimal control problem. The initial formulation, presented here from [12], required tuning of these constraints.

The second MPC formulation, which is the major contribution presented in this paper (alongside the required inverse kinematics reformulation), simplifies the controller by removing those smoothing constraints and instead tracks both states and inputs. This work includes simulations of each controller on different models of the spine, and compares the higher performance of the smoothing controller versus the lower tuning complexity of the input tracking controller (Table I.) This second design makes engineering of the control system much easier when applied to new tensegtry structures. These benefits are shown in sec. VIII, where the controller is applied to a new tensegrity spine with no change in the parameters.

**C. Paper Organization**

The following sections first introduce the system model of the spine, with the topology and dynamics (sec. II) then inverse kinematics (sec. IV.) The control goals are presented in sec. III, the two controllers are presented in sec. V, then are simulated with the models in sec. VI and VII. An additional controller simulation with a different, but similar, spine robot is shown in sec. VIII, illustrating the generality of the reference input tracking controller.

This work uses a three-dimensional model of the spine, with multiple vertebrae, for evaluating the smoothing controller (from [12].) Meanwhile, a reduced-order 2D model is used for the reference input tracking controller, with only a single moving vertebra. Although the improved input-tracking controller is tested on a simpler system than the previous smoothing controller, the tracked states of the vertebrae are the same. Therefore, the results are compared quantitatively in sec. VII, and the limitations of this comparison are discussed in sec. IX.

**D. Notation**

In this work, bold math such as \( \xi \) is used for vector or matrix quantities. Superscripts with parentheses, such as \( \xi^{(i)} \), represent scalars that are indexed out of a vector. For example, \( \xi^{(1)} \) represents the first element of the vector \( \xi \). Superscripts without parentheses are either exponents, such as \( Q^i \), or represent special quantities such as \( A^c \) and \( A^d \) for continuous versus discrete time. Subscripts with parentheses, such as \( u_{(i,j)} \), represent vectors or moments between quantities \( i \) and \( j \). Variables that are subscripted without parentheses, such as \( p_j \) and \( p_m \), represent different quantities. In addition, the subscripts \( i, j, k, \) and \( h \) imply different variables for different values of the subscript. Subscripts without parentheses also indicate a discrete timestep when used with \( t \), such as \( u_t \). Parentheses after a symbol indicate a function, such as \( u(t) \). A bar indicates a reference state or input, e.g. \( \xi \) or \( \bar{u} \). Finally, \( I_s, 1_s, \) and \( 0_s \) represent the identity matrix, ones vector, and zeroes vector of size \( s \) respectively.

**II. SPINE DYNAMICS MODEL**

The dynamics of this spine, as adapted from the ULTRA Spine work [12], uses a state-space model consisting of rigid-body states for each vertebra. Two models are presented here, used for the different controller formulations: the first is a 3-dimensional, 3-vertebra system, the other is a 2D projection of one single vertebra. Both derivations treat the vertebra’s rigid body as a set of point masses existing at the nodes of the tensegrity structure, as justified in the literature [35], [20].

**A. Vertebrar Topology**

The topology of one vertebra of the spine is defined by the locations of its point masses within the rigid body. Fig. 2a and 2b show visualizations of these topologies for one vertebra, in both models.

For the 3D model, each vertebra consists of five point masses at \( \{a_1, \ldots, a_5\} \), with one at its center, and one at each end of its four connected ‘bars’ (Fig. 2a.) Each bar extends outward at a 30° angle relative to the X-Y plane, and is 15cm long. These dimensions correspond to an early hardware prototype of the robot [11], and are (in cm):

**TABLE I: Model-Predictive Control Formulations: Smoothing vs. Reference Input Tracking**

| Controller formulation | Tuning consts. | Time discr. | Dynamics formulation | Simulation setup | Max. Error | Refs. |
|------------------------|----------------|------------|----------------------|------------------|------------|-------|
| Smoothing Ref. Input Tracking | 14 | 1e^-3 sec. | Continuous approx. | 3 vertebra, 3D | < 0.5 cm | [12] |
| 5 | 1e^-5 sec. | Discr., w/affine trans. | 1 vertebra, 2D | See sec. VII-B | - | |

The two controller formulations presented in this paper have different benefits with respect to tuning and performance. The five tuning constants (column two) of the more general reference input tracking controller are straightforward to chose, since all have physical interpretations (e.g., minimum cable tension, vertebra anti-collision distance) or are common to many optimal control problems (e.g., the \( Q \) and \( R \) weighting matrices in eqn. (86-87), and MPC horizon length.)
certain coordinates (Fig. 2: Topology of spine models in both 3D and 2D. Point almost-horizontal cables holding the vertebrae apart. The 2D, 1b), with four vertical cables along the spine’s edges and four contains eight total cables between adjacent vertebrae (Fig. 2). B. Cable Connectivity

Specifically, the positions of these point masses are, in cm, almost-horizontal cables holding the vertebrae apart. The 2D, a

For the 2D model, each vertebra consists of four point masses: one at its center, and one at each end of its three bars (Fig. 2b). The dimensions and angles between the legs are the same as the 3D model; however, the mass of a4 is twice the others, in order to account for the 3D point masses a4 and a5 projecting onto the same point in the X-Z plane. Specifically, the positions of these point masses are, in cm, a

B. Cable Connectivity

Each model has a set of cables that connect adjacent vertebrae, shown in red in Fig. 1. The 3D, three-vertebra model contains eight total cables between adjacent vertebrae (Fig. 1b), with four vertical cables along the spine’s edges and four almost-horizontal cables holding the vertebrae apart. The 2D, single-vertebra model has four cables between the vertebrae (Fig. 1a). This totals to 24 cables for the 3D model, versus four cables for the 2D model. These models also include one non-moving vertebra each, which provides the anchor points for the lowest set of cables, but is not part of the dynamics.

C. State-Space Model

The dynamics derivation contains the translations and rotations that map the rigid body states onto point-mass positions. Since an analytical dynamics model is needed for the model-predictive controllers, prior work in the field including kinematics-only models [20], [36], [21] and numerical methods [35], [15], [18] could not be used.

The cables suspending the vertebrae are treated as the control input to the system (described in sec. II-D.) Full knowledge of the system states at each timestep is assumed. Therefore, the continuous-time system model is of the form g(ξ, u) (3)

where g(ξ, u) is the nonlinear dynamics function, ξ ∈ R36 in 3D or R6 in 2D is the state vector, consisting of the rigid body states of all vertebrae in the spine, and u ∈ R24 in 3D or R4 in 2D is the input vector of the same dimension as number of cables.

The system state ξ contains the positions and rotations of each vertebra (as Euler angles), followed by their first order derivatives. For example, in the 3D model, for vertebra j, the coordinates (xj, yj, zj, θj, γj, φj) are elements 1 through 6 of each set of 12 states (per vertebra) within ξ. The topology of each vertebra is then expressed symbolically using these system states by rotating the local frame of a vertebra about the Euler angles in ξ, then translating by rigid body position.

During bending, the spine vertebrae experience relatively small rotations, so the Euler angles in the rigid body states do not need to be explicitly constrained to lie between [0, 2π].

For the 3D spine, denoting Rx, Ry, Rz as the 3D rotation matrices for each Euler angle, and with ex, ey and ez as the unit vectors for each Cartesian direction, the positions of each of the a1...a5 point masses are

\[
\begin{bmatrix}
q_{1j} \\
q_{2j} \\
q_{3j} \\
q_{4j} \\
q_{5j}
\end{bmatrix} = 
\begin{bmatrix}
x_j \hat{e}_x + y_j \hat{e}_y + z_j \hat{e}_z \\
x_j \hat{e}_x + y_j \hat{e}_y + z_j \hat{e}_z \\
x_j \hat{e}_x + y_j \hat{e}_y + z_j \hat{e}_z \\
x_j \hat{e}_x + y_j \hat{e}_y + z_j \hat{e}_z \\
x_j \hat{e}_x + y_j \hat{e}_y + z_j \hat{e}_z
\end{bmatrix} \in \mathbb{R}^{15} (4)
\]

Thus, the positions of all 15 point masses (in 3D) have been defined in terms of the 18 position/angle states in ξ. The 2D model removes the y, θ, φ coordinates, and only has four point masses, but is otherwise expressed in the same manner.

D. Cable Model as System Inputs

The cables within the spine apply forces between the vertebrae, with each cable treated as a virtual spring-damper. The control inputs are the rest lengths of the spring, as if a motor was to retract or extend a cable in the physical robot.
The distances between cable-connected nodes are stored as scalars \( l_i \), each as a function of the state system \( \xi \). These distances are calculated using the positions of the point masses \( (q_{kj}) \) from eqn. (4). Defining \( S \) as the total number of cables \( (S = 24 \text{ in 3D, or } S = 4 \text{ in 2D}) \), for \( i = \{1...S\} \), the scalar tension force on cable \( i \) is:

\[
F_i = k(l_i - \rho_i) - c \dot{l}_i \in \mathbb{R}
\]

where \( \rho_i \) is the rest length of cable \( i \). The spring constant \( (k = 2000 \, \text{N/m}) \) and damping constant \( (c = 100 \, \text{Ns/m}) \) are chosen from a prototype of the physical robot. This model assumes that the rest lengths of the cables can be controlled directly, as in:

\[
u = [\rho_1 \rho_2 \ldots \rho_S]^T \in \mathbb{R}^S
\]

The applied cable forces \( F_i \) must be constrained to be strictly non-negative; i.e., the cables cannot "push." The following adaptation of the dynamics equations allows for this constraint. When solving the equations below, the cable tensions are kept as a separate symbolic variable. Calculating \( g(\xi, u) \) becomes a three-step process: first, the cable tensions are calculated first given the equations above, then those tension values are rectified if they are negative:

\[
F_i = \begin{cases} F_i, & \text{if } F_i \geq 0 \\ 0, & \text{if } F_i < 0 \end{cases}
\]

where this correction occurs as a separate function in MATLAB. Finally, these rectified values are substituted in for \( F_i \) below in the solved dynamics.

The continuous-time function \( g(\xi, u) \) can be symbolically solved by substituting eqns. (4-6) into Lagrange’s equations. These models have \( S \) total cables, \( J \) vertebra(e), and \( K \) point masses per vertebra. The point masses weigh \( m = 0.026 \, \text{kg} \) each. This evenly distributes the 0.13 kg mass of one vertebra, as motivated by an early physical prototype of the robot. Here, \( q_{kj}^{(z)} \) is the Cartesian coordinate in the z-direction (i.e. the third element in 3D or second element in 2D) for point mass \( q_{kj} \) from eqn. (4), with \( g \) as the gravitational constant, and where \( \xi^{(d)} \) is the \( d \)-th element of the state vector. Lagrange’s equations then become:

\[
T = \frac{1}{2} \sum_{j=1}^{J} \sum_{k=1}^{K} m||\dot{q}_{kj}||^2
\]

\[
V = \sum_{j=1}^{J} \sum_{k=1}^{K} mgq_{kj}^{(z)}
\]

\[
L = T - V
\]

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{\xi}^{(d)}} - \frac{\partial L}{\partial \xi^{(d)}} = \sum_{i=1}^{S} \frac{\partial F_i}{\partial \xi^{(d)}}
\]

Eqn. (11) is a vector equality, with one equation for each index \( d \). The indexes for \( d \) are the position and angle states in the state vector. For the 2D model, these indices are the first three of the six states, \( d = \{1...3\} \), and for the 3D model, these indices are for each vertebra, as in \( d = \{1..6, 13..16, 25..30\} \).

There is one equation in (11) for each position and angle states of the entire robot, i.e., the dimension of eqn. (11) is half the size of the state vector. The constants for each of summations in (8-11) are summarized in Table II for both models.

The right-hand side and left-hand side of (11) are solved symbolically, then equated, and the derivatives of each state are solved for (since each \( q_{kj} \) is expressed in terms of \( \xi \)). In this formulation, the \( F_i \) cable forces are kept as a separate symbolic variable, and are not included in eqn. (9), instead appearing in the right-hand side of eqn. (11).

This is performed using MATLAB’s solve functionality. Software is available, written by the authors, that symbolically calculates the function \( g(\xi, u) \) in continuous time.

### III. Reference State Trajectories

In this work, the desired trajectory for the spine robot is a bending motion in the \( X-Z \) plane, consisting of translations and rotations for each moving vertebra. This trajectory is motivated by prior work [11], where the forward kinematics for this spine were used to determine the vertebra positions when one set of its vertical cables were retracted. As no a priori dynamic trajectories were available for this model, the controllers in sec. V do not include the tracking of vertebrae velocities. Consequently, this trajectory is not guaranteed to be dynamically feasible; however, it has been observed as the output of previous dynamics simulations, and is therefore judged as a reasonable control goal.

In the desired trajectory, the spine initializes in an upright position, then rotates counterclockwise about the global Y-axis by an angle \( \beta_j(t) \) for vertebra \( j \) at time \( t \), while being constrained to the \( X-Z \) plane (Fig. 3). The trajectory is discretized according to the number of timesteps for the simulation, and is linearly spaced between \( \beta_j(0) = 0 \) and \( \beta_j^{\text{max}} \).

As per prior work [11], [12], the spine vertebrae are separated by 10cm vertically in their starting positions. This is chosen based on the specifications of a physical prototype. Therefore, counting the first moving vertebra as \( j = 1 \), the initial heights of the vertebrae can be defined (in meters) as:

\[
\bar{z}_j(0) = 0.1j
\]

These initial heights also define the radius of the rotation:

\[
r_j = \bar{z}_j(0)
\]

Consequently, the reference positions of each vertebra over time, \( \bar{x}_j(t) \) and \( \bar{z}_j(t) \), are:

\[
\bar{x}_j(t) = r_j \sin(\beta_j(t)), \quad \bar{z}_j(t) = r_j \cos(\beta_j(t))
\]

\[1\text{https://github.com/BerkeleyExpertSystemTechnologiesLab/ultra-spine-simulations} \]
In addition, the desired rotation $\gamma_j(t)$ of each vertebra about its inertial Y-axis is defined to be the same as the sweep angle $\beta_j(t)$ for that vertebra. This keeps the Z-axis of each vertebra's local frame aligned with the vector $r_j$:

$$\gamma_j(t) = \beta_j(t)$$ (14)

For consistency with kinematics simulations in prior work [11], a maximum sweep angle $\beta_j^{\text{max}}$ is defined for each vertebra. The 2D model with one vertebra only uses $\beta_1^{\text{max}}$.

$$[\beta_1^{\text{max}}, \beta_2^{\text{max}}, \beta_3^{\text{max}}] = \left[ \frac{\pi}{16}, \frac{\pi}{12}, \frac{\pi}{8} \right]$$ (15)

All other states not mentioned above are set to zero in $\xi$.

The number of points in the state trajectory for the two controllers presented below are $n = 80$ for the smoothing controller, and $n = 400$ for the reference input tracking controller. These trajectory lengths correspond to durations of 80ms and 4ms for the two simulations respectively. (See Table I for simulation timesteps.) For the reference input tracking controller, this also required 400 input reference points, as calculated from the inverse kinematics in the following section.

The velocities implied here violate some assumptions in the following section for the 2D model. These assumptions and their implications are discussed in sec. IX.

IV. INVERSE KINEMATICS AS REFERENCE INPUT TRAJECTORIES

The second of the two controllers presented here tracks a reference input trajectory in addition to a reference state trajectory. In general, model-based controller formulations for trajectory tracking require both state and input equilibrium points, e.g., $\xi_i^{eq}$ and $u_i^{eq}$ at each timestep in discrete time ([37], ch. 7.5). Equilibrium inputs were unavailable for the first controller formulation, thus requiring the addition of smoothing terms to compensate. The use of a $u_i^{eq}$ trajectory for $\bar{u}$ in a controller, as calculated in this section, demonstrates the first instance of combining state and input reference tracking on a tensegrity spine robot.

A. Inverse Kinematics as Equilibria

The reference input trajectory in this work is found by solving an inverse kinematics problem for each state in the reference state trajectory. In general, this is not the same as an equilibrium point for the system in motion, particularly for highly dynamic motions such as those simulated in sec. VII.

However, inverse kinematics problems are simpler to formulate and calculate, and if the robot is designed to move quasi-statically over a long control horizon, represent an approximation to dynamic equilibria. Future controllers for tensegrity spine robots may use inverse dynamics in place of inverse kinematics.

B. Tensegrity Inverse Kinematics Prior Work

For tensegrity systems, the term ‘inverse kinematics’ has been used to refer to solving for a sequence of cable rest lengths (inputs to the system model) that hold rigid bodies in a sequence of equilibrium states [20]. The calculation for each single point in the sequence becomes a statics problem, and can be solved using force and moment balances on all rigid bodies. This usage is similar to traditional robotics applications, which use the term ‘inverse kinematics’ to describe the finding of joint forces and angles that position a robot arm in a specific location and orientation.

The introduction of the term ‘inverse kinematics’ for tensegrity robots appears in [20], which employs the force-density method for these calculations. The force-density method for solving structural statics problems has been extensively applied to tensegrity systems in the context of form-finding [38], [39], [40], since it transforms the set of nonlinear force and moment balance equations into a linear system. The general form of the force density method is presented below, which is then adapted such that solutions exist for the robot model in this work.

C. Force Density Method for Tensegrity Networks

The force density method for tensegrity systems, as networks of force-carrying structural members in tension or compression, assumes the following. These assumptions will be used in all formulations of inverse kinematics presented here. Since this algorithm is only used for the 2D spine controller, the problem is formulated in 2D.

- The tensegrity structure is represented by a set of nodes, for which all x and z coordinates are known.
- Structural members only exist as connections between nodes.
- All members in compression do not deform.
- Forces are only exerted at nodes.
Assume the tensegrity structure has \( s \) members in tension (cables), \( r \) members in compression (bars, or parts of a rigid body), \( n \) nodes, in a \( d \)-dimensional space. For the two dimensional (\( d = 2 \)) single-vertebra structure presented here, the nodes are sets of \( \{ a_1, \ldots, a_4 \} \) as in eqn. 2 and Fig. 2b for both the non-moving vertebra and for the moving vertebra, for a total of \( n = 8 \). The robot has \( s = 4 \) cables (red lines in Fig. 1a), and 3 rigid members per vertebra (gray bars in Fig. 2b) for a total of \( r = 6 \).

A connectivity matrix \( C \in \mathbb{R}^{(s+r)\times n} \) can be written that describes which nodes are connected by members, where the first \( s \) rows of \( C \) are assumed to correspond to cable members and the last \( r \) rows correspond to bar members. This matrix \( C \) is defined using a graph structure, where if member \( k \in \{1, \ldots, (s+r)\} \) connects nodes \( i \) and \( j \), then \( i \)-th and \( j \)-th columns in \( C \) are set to 1 and -1 respectively for row \( k \), as in

\[
C^{(k,i)} = 1, \quad C^{(k,j)} = -1
\]  

(16)

All other entries in \( C \) are 0. The connectivity matrix for the spine robot in this research is

\[
C = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]  

(17)

Then, for a \( d=2 \) structure in the \( X-Z \) plane, define the vectors

\[
\{ x, z, p_x, p_z \} \in \mathbb{R}^{4n}
\]  

(18)

where \( x, z \) are the coordinates for each of the \( n \) nodes, and \( p_x, p_z \) are the sum of the external forces applied at each node.

For the robot in this work, \( x \) and \( z \) are calculated by translating and rotating the \( \{ a_1, \ldots, a_4 \} \) frame from eqn. (2) according to the reference system state \( \xi_t \) at a given timestep. The external forces \( p_x \) and \( p_z \) are the sum of gravitational forces at all point masses (-\( mg \) at each of the \( n \) nodes) and any reaction forces at the nodes.

The nodal reaction forces for this particular model are obtained by assuming that the spine robot is sitting on a flat surface, such that nodes 2 and 3 (e.g., \( a_2 \) and \( a_3 \)) have pin-joint contacts with the ground, and that reaction forces \( R_{2x}, R_{2z}, R_{3x}, R_{3z} \) are present at those nodes. Here, \( R_{2x} \) and \( R_{3z} \) are indeterminate due to the symmetry of the structure, and are ignored.

The external force/moment balance when treating the tension network as one whole is the form of \( AR = b_R \) below in eqn. (20). Here, define \( v_{(i,j)} \in \mathbb{R}^d \) are the position vectors between node \( i \) and \( j \), picked out coordinate-wise from \( x \) and \( z \) for \( d = 2 \) dimensions, as in:

\[
v_{(i,j)} = \begin{bmatrix}
x^{(j)} - x^{(i)} \\
z^{(j)} - z^{(i)}
\end{bmatrix} \in \mathbb{R}^2
\]  

(19)

Noting that the first index of the vector, as in \( v^{(1)}_{(i,j)} \), is its x-component, and by summing moments around node 2 with the centers of mass of the two vertebrae at nodes 1 and 5, the external force/moment balance is

\[
\begin{bmatrix}
1 & 1 & 0 & v^{(1)}_{(2,3)} \\
0 & v^{(1)}_{(3,5)} & R_{2x} & R_{3x}
\end{bmatrix} = \begin{bmatrix}
8mg \\
4mg(v^{(1)}_{(2,3)} + v^{(1)}_{(3,5)})
\end{bmatrix}
\]  

(20)

Eqn. (20) is solved prior to the inverse kinematics problem by taking \( R = A_R^{-1} b_R \). The solutions to \( R_{2x} \) and \( R_{3x} \) are inserted into \( p_z \) as external reaction forces.

Finally, define the force density vector \( q \) as

\[
q = [q_1, q_2, q_3, \ldots, q_{s+r}]^T \in \mathbb{R}^{s+r}
\]  

(21)

such that if member \( k \) holds a force \( F_k \) along its length of \( l_k \),

\[
q_k = \frac{F_k}{l_k}
\]  

(22)

This variable \( q \) represents a different quantity than in sec. II, and is used here for consistency with the literature.

As seen in [20], [38], [39], [40] the force balance condition for static equilibrium of the structure can be stated as

\[
C^T \text{diag}(q) C x = p_x
\]  

(23)

\[
C^T \text{diag}(q) C z = p_z
\]

As also discussed in [20], [38], [40], eqn. 23 can be reorganized as

\[
A q = p,
\]  

(24)

\[
A = \begin{bmatrix}
C^T \text{diag}(C x) \\
C^T \text{diag}(C z)
\end{bmatrix} \in \mathbb{R}^{nd \times (s+r)}
\]  

(25)

\[
p = \begin{bmatrix}
p_x \\
p_z
\end{bmatrix} \in \mathbb{R}^{nd}
\]  

(26)

Here, \( A \) and \( p \) are constants at timestep \( t \), since each only depends upon the \( \xi_t \) constant at that timestep. Therefore, eqn. (24) is a set of linear equations in \( q \).

Solving for a specific \( q \) vector of force densities can be done by optimizing for the minimum total force density. Adding a constraint that each tension member has a minimum force density of \( c \in \mathbb{R} \), this optimization problem becomes:

\[
\min q^T q \\
\text{s.t. } A q = p \\
q_s - 1^c \geq 0
\]  

(27-29)

where

\[
q_s = [q_1, q_2, \ldots, q_s]^T \in \mathbb{R}^s
\]  

(30)

is defined to be the force densities for the cables only, and \( 1_s \in \mathbb{R}^s \) is a column vector of ones.

The optimization problem of eqn. (27-29) is the standard form of tensegrity inverse kinematics as used in [20], [21]. If a feasible \( q \) is found, the rest lengths \( \rho_i \) of each cable \( i \in
\{1, \ldots, s\} \) are back-calculated from eqn. (5) using the optimal force density \( q^* \), with \( k \) as the spring constant:
\[
\rho_i = l_i - l_i q_i^* \kappa_i
\]

(31)

D. Existence of Solutions and Rank Deficiency

However, issues arise with the existence of solutions to eqn. (24). In works such as [20], [11], [21], the tensegrity structure has many more cables and bars than nodes, such that \((s + r) > (nd)\). Thus, \( A \) is wider than it is tall, with a null space dimension of at least \((s + r) - (nd) > 0\), so an infinite number of solutions exist to eqn. (24). However, in this work, \((s + r) = 10\) and \((nd) = 16\), so \( A \) is taller than it is wide. The \( A \) matrix is full rank for this robot (\( \text{rank}(A) = 10 \)), and no solutions exist to eqn. (24).

This rank deficiency issue for static equilibrium is discussed in the literature on tensegrity structures in the context of geometry [41] and energy methods [42]. Algorithms exist for determining if a structure would have static equilibrium solutions [43], [26]. However, addressing this issue usually consists of adding cables or changing the geometry of the tensegrity structure itself (via form-finding, e.g. [44]), which is not possible given the control problem statement in this work.

Previously published work on reformulating the optimization problem of eqn. (27-29), in the context of robotics and control systems, did not address these rank issues when implemented for this 2D spine. Reducing (27-29) to a cable-only formulation by optimizing only over \( q_s \) as suggested in [20] only exacerbates these rank issues by defining \( A \) with fewer columns. Additionally, the relaxation of this equality-constrained problem to an inequality-constrained formulation, as used in [20], did not make the problem feasible.

E. Rigid Body Reformulation of the Force Density Method

This work adapts the node-graph formulation of the force density method (eqn. 23-29) by combining nodes according to their connections within a rigid body and neglecting the (internal) stresses between those nodes, and is the first time such a reformulation is presented. The reformulation admits solutions for the spine robot in this research. This process is described below as a “rigid body reformulation,” as in the context of rigid body mechanics where stresses within a body are neglected, although prior statics work uses the term ‘rigid’ in different contexts [41].

The following derivation is specific to the spine robot in this work, and has yet to be confirmed on larger tensegrity structures with more than two rigid bodies or in three dimensions. This derivation also assumes the following conditions in addition to those for the node-graph formulation:

- The sets of connected bars in the structure are attached as \( b \) rigid bodies, for which internal forces are neglected.
- All rigid bodies have the same number of compression members (bars).
- The nodes in the connectivity matrix are organized by rigid body, such that \( C \) is block-structured for the last \( r \) rows of compression members.

The first and second assumptions divide the \( n \) nodes and \( r \) bars of the entire structure into \( \eta = \frac{n}{s} \) nodes per rigid body and \( \lambda = \frac{r}{b} \) bars per rigid body. This is similar to the repeated ‘cells’ of a larger tensegrity, as the term is used in [44]. For this spine with \( n = 8 \) and \( b = 2 \), \( \eta = 4 \) and \( \lambda = 3 \). The third assumption can be seen for this setup in the \( C \) matrix of eqn. (17), where the last \( r \) rows of \( C \), e.g. rows 5-10, are block-diagonal in blocks of \( \lambda \times \eta \). Also implied here is that the block of \( s \) rows of cables apply forces to both rigid bodies.

The vector \( p \) of external forces (eqn. 26) can be collapsed from the force for each node into the sum of forces for each rigid body. Denoting this sum as \( p_f \),
\[
p_f = (I_{db} \otimes \mathbf{1}_\eta) p \in \mathbb{R}^{db}
\]

(32)

where \( I_{db} \in \mathbb{R}^{db \times db} \) is the identity matrix, \( \mathbf{1}_\eta \in \mathbb{R}^\eta \) is a column vector of ones, and \( \otimes \) is the Kronecker product. The \( I_q \) vector collapses all nodes per rigid body, and the \( I_{db} \) matrix is used to collapse the forces for both of the \( d = 2 \) dimensions for the \( b \) rigid bodies.

The lengths of each of the members can be calculated using the connectivity matrix and node positions, as in:
\[
d_x = C x, \quad d_z = C z \quad \in \mathbb{R}^{(s+r)},
\]

(33)

Then, only considering the cable force densities \( q_s \in \mathbb{R}^s \) as in eqn. (30), the force balance (without moments) for this tensegrity spine can be written as
\[
A_f q_s = p_f
\]

(34)

where \( A_f \) consists of the lengths of the \( s \) cable members, taken element-wise from \( d_x \) and \( d_z \). For \( b = 2 \) rigid bodies, the \( A_f \) matrix is
\[
A_f = \begin{bmatrix}
-d^{(1)}_x & -d^{(2)}_x & \cdots & -d^{(s)}_x \\
-d^{(1)}_z & -d^{(2)}_z & \cdots & -d^{(s)}_z \\
-d^{(1)}_x & -d^{(2)}_x & \cdots & -d^{(s)}_x \\
-d^{(1)}_z & -d^{(2)}_z & \cdots & -d^{(s)}_z
\end{bmatrix}
\in \mathbb{R}^{db \times s}
\]

(35)

For structures with \( b > 2 \) rigid bodies, where sets of cables apply forces to multiple bodies, or \( d = 3 \) dimensions, the \( A_f \) matrix will have a more complicated block structure.

A moment balance for both rigid bodies is also required now, since the point-mass formulation has been transformed into a rigid body formulation. First, consider the moments that arise from cable forces. Assuming cable \( k \) connects nodes \( i \) and \( j \), and using the position vector notation from eqn. (19), these cable forces \( F_k \) can be expressed in a vectorized form of eqn. (22) as
\[
F_k = v_{(i,j)} q_k \in \mathbb{R}^d
\]

(36)

Additionally, assume that the rigid bodies are symmetric, as is the case with these spine vertebrae, with node \( h \) representing the location of the center of mass of body \( b \). The moment
contribution from cable \( k \) (in a general \( d=3 \) dimensional formulation) for rigid body \( b \) is then
\[
M_{(b,k)} = \mathbf{v}_{(h,i)} \times \mathbf{F}_k = (\mathbf{v}_{(h,i)} \times \mathbf{v}_{(i,j)}) q_k \tag{37}
\]

Here, for body \( b=1 \), the center of mass is node \( h = 1 \), and for body \( b = 2, h = 5 \).

In \( d = 2 \) dimensions, as for this problem, the equivalent of the cross-product is a determinant, and moments are scalar. Denoting \( \det = | \cdot | \), eqn. (37) then becomes
\[
M_{(b,k)} = |\mathbf{v}_{(h,i)} \mathbf{v}_{(i,j)}| q_k \in \mathbb{R} \tag{38}
\]

The moments due all all cables on both of the \( b=2 \) rigid bodies for this spine are then \( A_m q_s \), where
\[
A_m = \begin{bmatrix}
M_{(1,1)} & \cdots & M_{(1,s)} \\
M_{(2,1)} & \cdots & M_{(2,s)}
\end{bmatrix} \in \mathbb{R}^{b \times s} \tag{39}
\]

As with the force balance in eqn. (35), if \( b > 2 \) rigid bodies, the \( A_m \) matrix will have a more complicated block structure.

External forces from \( p \) also contribute moments, though only certain components of the \( p \) vector appear here. In particular, reactions \( R_{2z} \) and \( R_{3z} \) act at nodes 2 and 3 on body 1, for which the center of mass is node \( h = 1 \). However, The point masses in the original formulation do not contribute to the moment balance in this rigid body reformulation, as the mass of the entire vertebra is lumped at its center of mass.

Writing the moment balance as
\[
A_m q_s = p_m, \tag{40}
\]

and since the moment arm is the \( x \)-component of the nodes’ position vectors, e.g. \( \mathbf{v}_{(h,i)}^{(1)} \) as in eqn. (20), the external moments are (with \( h = 1 \))
\[
p_m = \begin{bmatrix}
\mathbf{v}_{(h,2)}^{(1)} R_{2z} + \mathbf{v}_{(h,3)}^{(1)} R_{3z} \\
0
\end{bmatrix} \in \mathbb{R}^b. \tag{41}
\]

The force and moment balance conditions, eqns. (34) and (40), can then be combined by stacking the systems of equations, as in
\[
A_b = \begin{bmatrix}
A_f \\
A_m
\end{bmatrix}, \quad p_b = \begin{bmatrix}
p_f \\
p_m
\end{bmatrix}, \tag{42}
\]

so that the full static equilibrium condition is
\[
A_b q_s = p_b \tag{43}
\]

Future work will calculate the moment balance using the \( C \) matrix, will calculate the center of mass of each rigid body without assumptions of symmetry, and will provide a general method for more than \( b = 2 \) bodies in three dimensions. Although the algorithm would be more general, the approach provided here would still apply.

For the spine robot in this research, eqn. (43) has solutions. The matrix \( A_b \in \mathbb{R}^{8 \times 4} \) has rank 3, thus a null space of dimension \( 4 - 3 = 1 \). So, even though \( A_b \) is still wider than it is tall, the rigid body formulation admits solutions here whereas the node-graph formulation does not.

This rigid body formulation (eqn. 32-43) can be combined with the minimum force density optimization problem (eqn. 27-29) by substituting the equality constraints, as in
\[
\begin{align*}
\min_{q^s} & \quad q^T \mathbf{q}_s \\
\text{s.t.} & \quad A_b \mathbf{q}_s = p_b \tag{44} \\
& \quad q_s - 1, c \geq 0 \tag{45}
\end{align*}
\]

Problem (44-46) can be solved for each position in the reference state trajectory, as \( A_b \) and \( p_b \) vary with \( \xi_t \), using a quadratic programming solver.

**F. Relaxed Problem Formulation**

Finally, the optimization problem (44-46) can be relaxed from an equality constrained problem to an inequality constrained problem as suggested by [20] and as in the quadratic programming literature. This reformulation is automatically performed by higher-level solvers such as YALMIP [45] and MATLAB’s quadprog, so is derived here primarily for use in control settings with faster, lower-level solvers.

Following [20], all solutions to \( q_s \) in eqn. (43) can be expressed using the Moore-Penrose pseudoinverse, denoted \((\cdot)^+\), as in
\[
q_s = A_b^+ p_b + (1 - A_b^+ A_b) w \tag{47}
\]

where \( w \in \mathbb{R}^s \) is a free variable that allows the solver to search the null space of \( A_b \). The optimization is then in terms of \( w \). By defining
\[
V = (1 - A_b^+ A_b), \tag{48}
\]

for convenience, the objective function of the optimization problem, eqn. (44), becomes
\[
q^T \mathbf{q}_s = (A_b^+ p_b + V w)^T (A_b^+ p_b + V w), \tag{49}
\]

and by dropping terms that do not contain \( w \), the full optimization problem statement becomes
\[
\begin{align*}
\min_w & \quad w^T V^T V w + 2 w^T V A_b^+ p_b \\
\text{s.t.} & \quad A_b^+ p_b + V w - 1, c \geq 0 \tag{50} \\
& \quad w \in \mathbb{R}^s \tag{51}
\end{align*}
\]

Once the optimal solution to eqns. (50-51), denoted \( w^* \), is found using a solver, the inverse kinematics inputs for the controller can be calculated in a similar way as eqn. (31). Specifically, for cable \( i \in \{1, \ldots, s\} \), the rest length becomes
\[
q_i^* = (A_b^+ p_b + V w^*)^T, \quad \rho_i = l_i - \frac{l_i q_i^*}{k} \tag{52}
\]

**G. Implementation**

For the controller with inverse kinematics reference inputs described below, problem (50-51) was solved offline for each \( \xi_t \) using MATLAB’s quadprog solver. Then, \( \rho_i \) are calculated using (52), reorganized as a vector \( \mathbf{u}_t \) as in eqn. 6, and stored for each \( t \). These pairs of \( \xi_t \) and \( \mathbf{u}_t \) are then used as the reference trajectories.
V. MODEL-PREDICTIVE CONTROLLER FORMULATION

This work uses a model-predictive control (MPC) law for multiple reasons. First, there are inherent constraints on the dynamics of this system: the rest lengths of the springs cannot be negative, e.g., cannot have negative distance. In addition, the vertebrae of the spine should not collide. Finally, this MPC framework allows for the inclusion of smoothing constraints, used in one of the two controllers to enforce small changes between timesteps in the states and inputs.

The first controller formulation contains smoothing constraints and weighting terms, and is used on the 3D model. The second controller uses inverse kinematics as reference inputs, without smoothing terms, for the 2D spine. Neither formulation contains terminal constraints, and thus stability can only be shown experimentally, not proven.

A. Controller with Smoothing Constraints

A model-predictive controller is used to track the above trajectory $\xi$, for the 3-vertebrae, 3D model. At each timestep $t$ of the controller, the following constrained finite-time optimal control problem (CFTOC) is solved, generating the sequence of optimal control inputs $U^{t+1}_t = \{u^{t+1}_{1|t}, ..., u^{t+1}_{N|t}\}$, over a window of $N$ timesteps ahead. The notation $t + k|t$ represents a value at the timestep $t + k$, as predicted at timestep $t$ (from [46], Ch. 4.) The first input $u^{t}_{1|t}$ is applied, the system advances to $t + 1$, and the procedure is repeated, closing the loop as a model-predictive control law.

1) Constrained Finite-Time Optimal Control Problem Formulation: The following CFTOC problem is solved at each timestep $t$ using a quadratic programming solver.

$$\min_{U^{t+1}_t} p(\xi^{t+1}_t, \Delta \xi^{t+1}_t) \ldots$$

$$+ \sum_{k=0}^{N-1} q(\xi^{t+k|t}, \Delta \xi^{t+k|t}, \Delta u^{t+k|t})$$

s.t. $\xi^{t+k+1|t} = A_t \xi^{t+k|t} + B_t u^{t+k|t} + c_t$ (54)

$$\Delta \xi^{t+k|t} = \xi^{t+k|t} - \xi^{t+k-1|t}$$ (55)

$$\Delta u^{t+k|t} = u^{t+k|t} - u^{t+k-1|t}$$ (56)

$$\xi^{t|t} = \xi(t)$$ (57)

$$u^{min} \leq u^{t+k} \leq u^{max}$$ (58)

$$\|u^{t+k} - u^{t-1}\|_\infty \leq w_1$$ (59)

$$\|u^{t+k} - u^{t|t}\|_\infty \leq w_2, \quad k = 1..(N - 1)$$ (60)

$$\|u^{t+N|t} - u^{t|t}\|_\infty \leq w_3$$ (61)

$$\|\Delta \xi^{(1:6)}_{t+k|t}\|_\infty \leq w_4$$ (62)

$$\|\Delta \xi^{(1:18)}_{t+k|t}\|_\infty \leq w_5$$ (63)

$$\|\Delta \xi^{(25:30)}_{t+k|t}\|_\infty \leq w_6$$ (64)

$$\xi^{t+k|t} + w_7 \leq \xi^{(15)}_{t+k|t}$$ (65)

$$\xi^{(15)}_{t+k|t} + w_7 \leq \xi^{(27)}_{t+k|t}$$ (66)

Here, $N = 10$ is the horizon length (a scalar), $w_1 \ldots w_7$ are constant scalar weights, and $\xi_{t+k|t}^{(i)}$ denotes the $i$-th element of the state vector at time $t + k$ as predicted at time $t$. The functions $p$ and $q$ represent the terminal cost and stage cost of the objective function, as opposed to the inverse kinematics force balance of sec. IV. The following sections define the objective function, and use and purposes of the constraints.

2) Dynamics Constraint: The dynamics constraint (54) consists of the linearized system at timestep $t$, as in:

$$A_t = \frac{\partial g(\xi, u)}{\partial \xi} \bigg|_{\xi = \xi_{t-1}, u = u_{t-1}}$$ (67)

$$B_t = \frac{\partial g(\xi, u)}{\partial u} \bigg|_{\xi = \xi_{t-1}, u = u_{t-1}}$$ (68)

$$c_t = g(\xi_{t-1}, u_{t-1}) - A_t \xi_{t-1} - B_t u_{t-1}$$ (69)

This linearization (67-69) is implemented as a finite difference approximation in MATLAB, using the equations of motion as solved from eqn. (11). This approach is chosen due to computational issues with calculating additional analytical derivatives of the dynamics.

The linearization is calculated at each timestep $t$ and used for the optimization over the entire horizon, thus the notation $A_t, B_t, c_t$. For the start of the simulation, $u_0 = 0$ is used. Since these linearizations are not at equilibrium points, the linear system is affine, with $c_t$ being a constant vector offset.

In this control law, the continuous-time linearized dynamics are used as a constraint, and are not discretized. Since the timesteps are small ($dt = 0.001$ sec.), these representations are similar enough that the discretization step can be approximated away. The second controller in this work, with reference input tracking, removes this approximation.

3) Other Constraints: The remaining constraints are either smoothing terms, constraints motivated by the physical robot, or miscellaneous helper variables.

Constraints (55) and (56) define the $\Delta u$ and $\Delta \xi$ variables, which are used for the smoothing constraints on the inputs and states. Constraint (57) assigns the state variable at the start of the optimization horizon, $\xi_{t|t}$, to the actual observed value of the state from the previous simulation timestep, $\xi(t)$.

Constraint (58) is a bound on the inputs, limiting the length of the cable rest lengths, with $u^{min}, u^{max} \in \mathbb{R}^{24}$ but having the same value for all inputs (Table III).

Constraints (59-64) are smoothing terms to compensate for the lack of an input reference trajectory. Of these, (59-61) are for the inputs, where $u_{t-1}$ is the most recent input at the start of the CFTOC problem. Constraints (62-64) are smoothing terms on the states, limiting the deviation between successive states in the trajectory. These reduce linearization error, and are split so that the positions and angles of each vertebra could be weighted differently. No velocity terms are constrained.

Finally, since states $\{\xi^{(3)}, \xi^{(15)}, \xi^{(27)}\}$ are the vertebra z-positions, constraints (65-66) prevent vertebra collisions.
4) Objective Function: The objective function has two components, a terminal cost \( p \) and a stage cost \( q \). These are defined as the following. Here, the notation \( \| \Delta \xi_{t+k+1} \|_2^2 \) denotes a quadratic term on a vector weighted by a matrix, as in \((\Delta \xi_{t+k+1})^T S^k(\Delta \xi_{t+k+1})\).

\[
p(\xi_{t+N|t}, \Delta \xi_{t+N|t}) = \| \xi_{t+N|t} - \xi_{t+1|t} \|_Q^2 + \| \Delta \xi_{t+N|t} \|_S^2 \]
\[
g(\xi_{t+k+1|t}, \Delta \xi_{t+k+1|t}, \Delta u_{t+k|t}) = \| \xi_{t+k+1|t} - \xi_{t+k|t} \|_Q^2 + \| \Delta \xi_{t+k+1|t} \|_S^2 + w_s \| \Delta u_{t+k|t} \|_\infty \]
\]

As before, \( w_s \) is a scalar, while \( Q \) and \( S \) are constant diagonal weighting matrices which are exponentiated by the timestep in the optimization horizon. Here, \( Q \) penalizes the tracking error in the states, \( S \) penalizes the deviation in the states at one timestep to the next, and \( w_s \) penalizes the deviations in the inputs from one timestep to the next. These matrices are diagonal, with blocks corresponding to the Cartesian and Euler angle dimensions, with zeros for all velocity states, according to vertebra. Nonzero values are at states \( \xi^{(1)} \ldots \xi^{(N)} \), \( \xi^{(13)} \ldots \xi^{(18)} \), and \( \xi^{(29)} \ldots \xi^{(30)} \), recalling that \( \xi \in \mathbb{R}^{30} \).

Raising each diagonal element of \( Q \) or \( S \) to the power of \( k \) or \( N \) puts a heavier penalty on terms farther away on the horizon. These are defined as:

\[
\tilde{Q}^k = \text{diag}(w_9^k, w_9^k, w_9^k | w_{10}^k, w_{10}^k, w_{10}^k | 0) \in \mathbb{R}^{12 \times 12}
\]
\[
\tilde{S}^k = \text{diag}(w_{11}^k, w_{11}^k, w_{11}^k | w_{12}^k, w_{12}^k, w_{12}^k | 0) \in \mathbb{R}^{12 \times 12}
\]
\[
Q^k = \mathbf{I}_3 \otimes \tilde{Q}^k, \quad S^k = \mathbf{I}_3 \otimes \tilde{S}^k
\]

where, as with eqn. (32), the Kronecker product with \( \mathbf{I} \) is used as a shorthand to pattern a matrix along the main diagonal.

Table III lists all the constants for this controller, including the constraints and the objective function, with units.

| Constant | Value | Interpretation |
|----------|-------|----------------|
| \( N \)  | 10    | no units       |
| \( \bar{u}_{min} \) | 0.01 meters (cable) | Max. Cable Length |
| \( \bar{u}_{max} \) | 0.20 meters (cable) | Min. Cable Length |
| \( \bar{u}_1 \) | 0.01 meters (cable) | Input Smooth., Horiz. Start |
| \( \bar{u}_2 \) | 0.01 meters (cable) | Input Smooth., Horiz. Middle |
| \( \bar{u}_3 \) | 0.10 meters (cable) | Input Smooth., Horiz. End |
| \( \bar{u}_4 \) | 0.02 meters and radians | State Smooth., Bottom Vert. |
| \( \bar{u}_5 \) | 0.03 meters and radians | State Smooth., Mid. Vert. |
| \( \bar{u}_6 \) | 0.04 meters and radians | State Smooth., Top Vertebra |
| \( \bar{u}_7 \) | 0.02 meters (vertebrae pos.) | Vertebral Anti-Collision |
| \( \bar{u}_8 \) | 3 | no units | Input Difference Penalty |
| \( \bar{u}_9 \) | 25 | no units | State Tracking, Vertebr. Pos. |
| \( \bar{u}_{10} \) | 30 | no units | State Tracking, Vert. Angle |

B. Controller with Input Reference Tracking

One of the main contributions of this work (in comparison to [12]) is an MPC control law that tracks reference inputs. These are in turn calculated from the inverse kinematics of the tensegrity spine structure, from the previous section. As discussed in this section, the controller is formulated for the 2D, single-vertebra spine model. The structure of the closed-loop MPC law is the same as that in section V-A, so is not repeated here.

1) Constrained Finite-Time Optimal Control Problem Formulation: The following CFTOC problem is solved at each timestep \( t \) using a quadratic programming solver.

\[
\min_{\bar{u}_{t+\cdots+N|t}} \ p(\xi_{t+N|t}) + \sum_{k=0}^{N-1} g(\xi_{t+k|t}, u_{t+k|t})
\]
\[
\text{s.t.} \quad \xi_{t+k+1|t} = A_i \xi_{t+k|t} + B_i u_{t+k|t} + c_i \]
\[
\xi_{t|t} = \xi(t)
\]
\[
u_{t+k|t} \geq u_{min}
\]
\[
u_{t+k|t} \leq w_1
\]

This formulation (73-77) is significantly simpler than the smoothing formulation (53-66). Here, \( N = 4 \) is the horizon length (a scalar), \( w_1 \) is a constant scalar weight, and \( \xi_{t+k|t} \) denotes the \( i \)-th element of the state vector at time \( t + k \) as predicted at time \( t \). As above, \( p \) and \( q \) represent the terminal cost and stage cost of the objective function. The following sections define the objective function, and use and purposes of the constraints.

2) Dynamics Constraint: Constraint (74) enforces the system dynamics which are linearized and discretized at each timestep \( t \). As with the formulation of the smoothing controller, the linearization occurs as eqns. (67-69), with a finite difference approximation. However, unlike the above formulation, this controller also discretizes the linearized system, in order to reduce modeling error.

The linearized system comes in the form of

\[
\dot{\xi}_t = A_i \xi_t + B_i u_t + c_i
\]

from (67-69). Here, \( A_i \in \mathbb{R}^{6 \times 6} \), \( B_i \in \mathbb{R}^{6 \times 4} \), and \( c_i \in \mathbb{R}^6 \). In order to discretize the system, the affine term \( c_i \) is absorbed into a new augmented matrix \( A_i \) alongside the original \( A_i \). This technique allows the affine system to be re-written as a linear system with an augmented state vector, as in

\[
\dot{\tilde{\xi}}_t = \tilde{A}_i \tilde{\xi}_t + \tilde{B}_i u_t
\]

and, where \( \tilde{0}_i \) is a column vector of zeros of length \( i \),

\[
\tilde{A}_i = \begin{bmatrix} A_i & c_i \\ \phi_i & 1 \end{bmatrix}, \quad \tilde{B}_i = \begin{bmatrix} B_i \\ 0_i \end{bmatrix}, \quad \tilde{\xi}_t = \begin{bmatrix} \xi_t \\ 0_i \end{bmatrix}
\]

This linear system can then be discretized using a zero-order hold. The zero-order hold makes the assumption that the control input is constant between successive timesteps, e.g. that
\[ u(\tau) = u_t, \forall \tau \in [t, t + 1) \text{.} \] Denoting the discretized system matrices as \((\cdot)^d\), with a discretization timestep of \(T_s\) chosen to be the same as the \(dt\) of the controller, the zero-order hold can be calculated by transforming the linear system matrices:

\[
\begin{align*}
\tilde{A}^d_t &= e^{\tilde{A} t} T_s \\
\tilde{B}^d_t &= (\tilde{A}_t)^{-1}(\tilde{A}^d_t - I) \tilde{B}_t
\end{align*}
\] (81)  

(82)

The zero-order hold transformation here is derived for a linear system, not for the affine-augmented system as used. Through the course of testing this algorithm, it was confirmed that the \(\tilde{A}^d_t\) and \(\tilde{B}^d_t\) matrices maintain the same block structure as in eqn. (80), particularly for the rows of 0 and 1 terms, up to machine precision. Thus, it was concluded that discretized system could be expanded back out into an affine system by separating out each block. Specifically,

\[
\begin{align*}
\tilde{A}^d_t &= \tilde{A}_t^{d(1:6,1:6)} \\
\tilde{B}^d_t &= \tilde{B}_t^{d(1:6,1:4)} \\
\tilde{c}^d_t &= \tilde{A}_t^{d(1:6,7)}
\end{align*}
\] (83)

The linearized and discretized matrices in eqn. (83) are then used in eqn. (74), where the same linearization is applied over the \(N\)-step optimization horizon. The calculations (80-83) are performed at each timestep of the controller using MATLAB’s \(c2d\) command.

3) Other Constraints: The remaining constraints have the same interpretations as their counterparts in the smoothing controller formulation. Constraint (75) assigns the initial condition at the starting time of the CFTOC problem. Constraint (76) is a linear constraint on the inputs so that the cables cannot have negative rest lengths. Finally, constraint (77) denotes a minimum bound on the second element in the state, the \(z\)-position, which prevents collision between the moving vertebra and the static vertebra.

4) Objective Function: The objective function for this formulation is comprised of quadratic weights on the state and input tracking errors. Using similar notation as in equations (70) and (71),

\[
\begin{align*}
p(\xi_{t+N|t}|) &= ||\xi_{t+N|t} - \tilde{\xi}_{t+N|t}\|^2_Q \\
q(\xi_{t+k|t}, u_{t+k|t}) &= ||\xi_{t+k|t} - \tilde{\xi}_{t+k|t}\|^2_Q + ||u_{t+k|t} - \tilde{u}_{t+k|t}\|^2_R
\end{align*}
\] (84)  

(85)

Here, \(Q\) and \(R\) are constant diagonal weighing matrices which penalize state and input tracking errors respectively, defined similarly to the smoothing formulation, but do not vary with the horizon step as with the \(Q^k\) terms in eqn. (71). Specifically, these weights are

\[
Q = diag(w_2, w_2, w_2 | 0...0) \in \mathbb{R}^{6 \times 6}
\] (86)  

\[
R = diag(w_3, w_3, w_3, w_3) \in \mathbb{R}^{4 \times 4}
\] (87)

As with eqn. (72), the \(Q\) matrix does not penalize velocity states. Table IV lists all the constants for this controller, including the constraints and the objective function, with units.

**C. Differences between controller formulations**

The differences between the two controller formulations (sec. V-A and V-B) are summarized in table I. In addition to the inherent difference between the tracking of one vertebra versus 3 vertebrae, and the difference between the 2D and 3D models that are tracked, three major considerations are present.

First, the more general reference input tracking controller does away with the smoothing constraints and reduces the complexity of the CFTOC problem, thus removing most of the need for tuning optimization weights (compare table III versus table IV). Second, the more general controller includes a discretization of the affine dynamics of the system, reducing modeling error, although this improvement could also be implemented for the smoothing controller at the cost of some increase in computational load.

Third, in contrast to those benefits, the more general controller requires a faster simulation rate as tested here, with the discretization timestep of \(dt = 1 e^{-3}\) versus \(1 e^{-5}\) for the smoothing controller. This increase in frequency leads to an increase in trajectory points (\(n = 400\) simulation steps versus \(n = 80\)) to compensate for the shorter timestep between points, in order to have the vertebrae move at similar velocities. These three changes represent the tradeoffs between tuning requirements and computational performance implications of either controller.

**VI. SIMULATION SETUP**

Two main sets of simulations are presented in this work, one for the controller with smoothing constraints and one for the controller with input reference tracking. All simulation work used the YALMIP toolbox in MATLAB [45], with Gurobi as the solver [47]. The dynamics for the 3D system are forward-simulated using the Runge-Kutta method, whereas the dynamics for the 2D system are forward-simulated using a first-order Euler integration for computational efficiency. All code is available online, as with the dynamics solutions2.

**A. Noise Model**

For both models and controllers, simulations are also performed with noise, in order to test closed-loop performance. Process noise is implemented by adding a sample from a normally-distributed random variable to the system dynamics during the simulation. For example, denoting the timestep as \(\Delta t\), the forward-Euler-simulated model for the 2D spine is

\[
\xi_{t+1} = \xi_{t|t} + g(\xi_{t|t}, u_{t|t})(\Delta t) + \epsilon_t
\] (88)

2https://github.com/BerkeleyExpertSystemTechnologiesLab/ultra-spine-simulations
where $\epsilon_t$ is a sample drawn from $\epsilon \sim \mathcal{N}(0_2, I_2) \in \mathbb{R}^2$ at time $t$. The the weighting matrix $E$ scales the variance of the random variable. The simulation of the 3D model, using Runge-Kutta integration, works similarly. 

$E$ has different weights according to the position (and angle) states or the velocities, and is sized appropriately for either the 3D or 2D state model:

$$
E^{3D} = 13 \begin{bmatrix}
 w_{12} & 16 \\
 0_6 & w_{13} \\
 0_6 & 16
\end{bmatrix} \in \mathbb{R}^{36 \times 2} \quad (89)
$$

$$
E^{2D} = \begin{bmatrix}
 w_{14} & 13 \\
 0_3 & w_{15} \\
 0_3 & 13
\end{bmatrix} \in \mathbb{R}^{6 \times 2} \quad (90)
$$

Table V lists the values for $\{w_{12} \ldots w_{15}\}$. These values are selected such that the standard deviation of $\epsilon$ is scaled to roughly 33% of the maximum position displacement (or maximum velocity, respectively) of the robot between timesteps in the reference state trajectory.

| Constant: | Value: | Interpretation: |
|-----------|--------|-----------------|
| $w_{12}$  | $5 \times 10^{-4}$ | meters, rad. | Noise std. dev., positions/angles, 3D model |
| $w_{13}$  | $2 \times 10^{-4}$ | m/s, rad/s | Noise std. dev., velocities, 3D model |
| $w_{14}$  | $1.6 \times 10^{-5}$ | meters, rad. | Noise std. dev., positions/angles, 2D model |
| $w_{15}$  | $6.6 \times 10^{-6}$ | m/s, rad/s | Noise std. dev., velocities, 2D model |

### VII. RESULTS

The optimization problem for the smoothing controller, applied to the 3D model (from sec. V-A) took $0.5 \sim 1$ sec. to solve at each timestep, using the Gurobi solver. The optimization problem for the reference input tracking controller, applied to the 2D model (from sec. V-B), took $0.15 \sim 0.2$ sec. to solve at each timestep. The inverse kinematics inputs for the reference input tracking controller are calculated offline, so are not included in these statistics, but are of low enough computational load so as to be negligible in comparison.

Both controllers tracked the vertebrae states with sufficiently low error as to justify their use. The smoothing controller tracked with lower error, after an initial transient response, but had higher computational complexity and tuning requirements. The more general input-tracking controller exhibited lag, and thus larger tracking errors, but with lower computational overhead and with significantly less hand-tuning.

#### A. Controller with Smoothing Constraints

Fig. 4 shows the paths of the vertebrae in the 3D, three-vertebra simulation, using the smoothing constraint controller, in the $X$-$Z$ plane as they sweep through their counterclockwise bending motion. Fig. 4 includes the reference trajectory (blue), the resulting trajectory with MPC controller and no noise (green), and a representative result of controller with added noise (magenta). Fig. 5 shows a larger view of the top vertebra center of mass, which had the largest tracking errors of the three vertebrae, and which is used for comparison with the 2D single-vertebra model below.

#### B. Controller with Input Reference Tracking

Fig. 7 shows the path of the single vertebra in the 2D simulation, using the input reference tracking controller, as it sweeps through its counterclockwise bending motion. As with Fig. 4 and 5, the reference state trajectory is included (blue) alongside results from the controller with no noise (green) and from a representative simulation with noise (magenta.) The vertebra follows the path of the of the kinematic states, but experiences some accumulation of lag. The results show
Fig. 6: Tracking errors in system states for the 3D, three-vertebra model using the smoothing controller, with and without noise. Position states \((x, y, z)\) on the left with units of cm, Euler angles \((\theta, \gamma, \psi)\) on the right with units of degrees.

Fig. 7: Positions in the X-Z plane of the single vertebra for the 2D model, using with the reference input tracking controller, including the state reference and the two simulations (with/without noise), as the robot performs a counterclockwise bend. The vertebra tracks the trajectory, but accumulates more lag in comparison to the smoothing controller (Fig. 5.)

that the closed-loop controller is noise-insensitive, alongside accurate tracking, but that the lag occurs in all circumstances.

The tracking errors for each state are shown in Fig. 8, using the same convention as Fig. 6. The controller accumulates lag throughout the simulation, and the errors do not converge. This is expected, since the tracked inputs are inverse kinematics and not dynamics, and this simulation setup violates the assumption of quasi-static movement. Since the results presented here are used to compare with the smoothing controller, the simulations use the same setup with only dynamic movement. It is expected that given a setup where the controller has the opportunity to settle, the errors would converge.
VIII. CONTROL OF DIFFERENT SPINES

Since one of the primary advantages of the proposed input reference tracking controller is the lack of tuning parameters (in relative comparison to the smoothing controller), the new controller is easily extendable to different sizes and shapes of spines, whereas a large amount of tuning may have otherwise been required. In order to illustrate this, the controller was tested on a different 2D spine, with a different size and shape of vertebra. Control results (Fig. 9) show equivalent performance to the original vertebrae of sec. VII with no change in any tuning constants.

This differently-shaped spine still retained the same number of point masses, and is symmetric (to satisfy the assumptions of the inverse kinematics algorithm), but is now larger and heavier, with different angles between its bars. These changes are motivated by ongoing designs of hardware prototypes. The vertebra weighed a total of 0.2 kg, and the positions of its point masses (nodes) are, in cm,

\[
\begin{bmatrix}
a_1 & a_2 & a_3 & a_4 \\
0 & 20 & -20 & 0 \\
0 & -20 & -20 & 20
\end{bmatrix}
\]  

The reference state trajectory is adjusted to match the new size, with the same $\beta_{1}^{\text{max}}$ but a height and radius of $\bar{z}_1(0) = r_1 = 0.3$ m. Simulated noise is scaled in the same way. Accounting for the number of timesteps and distance traveled in comparison to the top vertebra of the 3D model, the weights for the noise are $w_{14} = 5e^{-4}$ and $w_{15} = 2e^{-4}$.

No changes were made to the inverse kinematics algorithm, nor to any of the constants in table IV. Fig. 9 shows similar performance to Fig. 7 despite the size and geometry change, illustrating the generality of the proposed controller.

IX. DISCUSSION

Both controllers exhibit state tracking characteristics which could be used in different environments for effective closed-loop control. In total, this is the first work (with [12]) that tracks a state-space trajectory of a tensegrity spine robot in closed-loop, and the first which shows noise insensitivity.

A. Computational Performance

The lengths of time taken to solve the optimization problem for each controller (0.5-1 sec. and 0.15-0.2 sec.) were longer than the timesteps of each respective simulation ($1e^{-3}$ and $1e^{-4}$ sec.). Thus, the optimization procedure will need to be made more efficient before using this controller in hardware. One approach that may reduce solver time is the calculation of a symbolic Jacobian for the $A_i$ and $B_i$ matrices, reducing the computation load in the linearization.

B. Tracking Performance Comparison

The tracking of an input trajectory removed the need for hand-tuned smoothing constraints, but exhibited lag in tracking a highly-dynamic state trajectory. This motivates the use of either controller in different settings. The smoothing controller may be appropriate for high-performance dynamic tracking, when the control system designer is able to tune the constraints. In contrast, the inverse kinematics input tracking controller may be appropriate for more pseudo-static movements, but can be implemented more reliably and on more systems without the tuned constraints.

C. Limitations Of Comparison

The results of tracking performance provided here uses the top vertebra of the 3D model in comparison to the single vertebra in the 2D model. This comparison is chosen to demonstrate the largest errors of each simulation. Thus, Fig. 5 and 7 represent the same geometry of state trajectory, but do not represent the exact same system model.

Though the input reference tracking controller is prototyped in a reduced-order version of the spine, the formulation is general enough to be readily implemented on a multiple-vertebra, 3D spine. However, such simulations have not been implemented, and as such, it is unknown if some combination of both optimization problems in sec. V-A and V-B may still be required for the higher-dimensional system.

Similarly, the input tracking controller required a very high frequency, which is unrealistic for hardware experiments without computational improvements. Such a high frequency arose from the linearization and discretization of the system model, and may not be required if the linearization is removed. Nonlinear MPC may address these challenges by removing the linearization step.

X. CONCLUSION

This work contributes two controller formulations and one inverse kinematics re-formulation for tensegrity robots, as well as simulations showing their efficacy on two models of tensegrity spines. The two controllers have different benefits, with higher performance of the smoothing formulation comparing against the the lower tuning complexity of the input tracking formulation. The second of the two controllers, which uses the inverse kinematics formulation for reference input tracking, shows tracking performance and noise insensitivity appropriate for use in quasi-static motions of these robots, and is shown to be sufficiently general to apply to multiple tensegrity spines without any tuning required.
Future work will focus on improved performance. Performance improvements, in addition to those mentioned above, may be achieved using inverse dynamics instead of inverse kinematics solutions. Tracking error may also be reduced by smoothing the hybrid dynamics of the cable slackness model, and if used together with nonlinear model-predictive control to remove the linearization, may allow for a lower-frequency controller to stabilize.

Finally, hardware experiments using such a lower-frequency controller may be conducted to evaluate the rejection of modeling error in addition to noise.

ACKNOWLEDGMENT

Many thanks to those who contributed to earlier versions of this work. In particular, thanks to Mallory C. Daly and Ellande Tang for early work on the inverse kinematics formulation, as well as Abishek K. Akella, Zeerak A. Ahmad, and Vytas SunSpiral for their contributions to the earlier conference version of this paper. Many thanks to Jeffrey Friesen, Kyunam Kim, Prof. Francesco Borrelli, and Prof. Andy Packard for feedback on this research.

REFERENCES

[1] Y. Fukuoka, H. Kimura, and A. H. Cohen, “Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts,” The International Journal of Robotics Research, vol. 22, no. 3-4, pp. 187–202, mar 2003.

[2] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “BigDog, the Rough-Terrain Quadruped Robot,” IFAC Proceedings Volumes, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[3] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, “Compliant quadruped locomotion over rough terrain,” in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, oct 2009, pp. 814–820.

[4] S. Maleki, A. Parsa, and M. N. Ahmadabadi, “Modeling, control and gait design of a quadruped robot with active spine towards energy efficiency,” in 2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM). IEEE, oct 2015, pp. 271–276.

[5] T. Horvat, K. Karakasilios, K. Melo, L. Fleury, R. Thandiacakkil, and A. J. Ispeert, “Inverse kinematics and reflex based controller for body-limb coordination of a salamander-like robot walking on uneven terrain,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, sep 2015, pp. 195–201.

[6] K. Weinmeister, P. Eckert, H. Witte, and A.-J. Ispeert, “Cheetah-cub-S: Steering of a quadruped robot using trunk motion,” in 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, oct 2015, pp. 1–6.

[7] P. Eckert, A. Sprowitz, H. Witte, and A. J. Ispeert, “Comparing the effect of different spine and leg designs for a small bouncing quadruped robot,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, may 2015, pp. 3128–3133.

[8] D. Hustig-Schultz, V. SunSpiral, and M. Teodorescu, “ Morphological design for controlled tensegrity quadruped locomotion,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Deajeon, Korea: IEEE, oct 2016, pp. 4714–4719.

[9] K. Miki and K. Tsujita, “A study of the effect of structural damping on gait stability in quadrupedal locomotion using a musculoskeletal robot,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, oct 2012, pp. 1976–1981.

[10] A. P. Sabelhaus, L. Janse van Vuuren, A. Joshi, E. Zhu, H. J. Garnier, K. A. Sover, J. Navarro, A. K. Agogino, and A. M. Agogino, “Design, Simulation, and Testing of Laika, a Quadruped Robot with a Flexible Actuated Spine,” in arXiv:1804.06527, 2018.

[11] A. P. Sabelhaus, H. Ji, P. Hylton, Y. Madaan, C. Yang, A. M. Agogino, J. Friesen, and V. SunSpiral, “Mechanism Design and Simulation of the ULTRA Spine: A Tensegrity Robot,” in ASME International Design Engineering Technical Conferences (IDETC) 39th Mechanisms and Robotics Conference. ASME, aug 2015.

[12] A. P. Sabelhaus, A. K. Akella, Z. A. Ahmad, and V. SunSpiral, “Model-Predictive Control of a flexible spine robot,” in 2017 American Control Conference (ACC). IEEE, may 2017, pp. 5051–5057.

[13] R. E. Skelton and M. C. de Oliveira, Tensegird systems. Springer, 2009.

[14] A. P. Sabelhaus, J. Bruce, K. Caluwaerts, P. Manovi, R. F. Firooz, S. Dobi, A. M. Agogino, and V. SunSpiral, “System design and locomotion of SUPERball, an unethered tensegrity robot,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, may 2015, pp. 2867–2873.

[15] K. Kim, A. K. Agogino, A. Toghyan, D. Moon, L. Taneja, and A. M. Agogino, “Robust learning of tensegrity robot control for locomotion through form-finding,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, sep 2015, pp. 5824–5831.

[16] C. Paul, F. Valero-Cuevas, and H. Lipson, “Design and control of tensegrity robots for locomotion,” IEEE Transactions on Robotics, vol. 22, no. 5, pp. 944–957, oct 2006.

[17] B. R. Tietz, R. W. Carnahan, R. J. Bachmann, R. D. Quinn, and V. SunSpiral, “Tetraspine: Robust terrain handling on a tensegrity robot using central pattern generators,” in 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, jul 2013, pp. 261–267.

[18] B. T. Mirletz, P. Bhandal, R. D. Adams, A. K. Agogino, R. D. Quinn, and V. SunSpiral, “Goal-Directed CPG-Based Control for Tensegrity Spines with Many Degrees of Freedom Traversing Irregular Terrain,” Soft Robotics, vol. 2, no. 4, pp. 165–176, dec 2015.

[19] T. Bliss, T. Iwasaki, and H. Bart-Smith, “Central Pattern Generator Control of a Tensegird Swimmer,” IEEE/ASME Transactions on Mechatronics, vol. 18, no. 2, pp. 586–597, apr 2013.

[20] J. Friesen, A. Pogue, T. Bewley, M. de Oliveira, R. Skelton, and V. SunSpiral, “DuCTT: A tensegird robot for exploring duct systems,” in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, may 2014, pp. 4222–4228.

[21] J. M. Friesen, P. Glick, M. Fenton, P. Manovi, A. Xydes, T. Bewley, and V. SunSpiral, “The second generation prototype of a Duct Climbing Tensegird Robot, DuCTT,” in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, may 2016, pp. 2123–2128.

[22] B. T. Mirletz, I.-W. Park, R. D. Quinn, and V. SunSpiral, “Towards bridging the reality gap between tensegird simulation and robotic hardware,” in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, sep 2015, pp. 5357–5363.

[23] J. Aldrich, R. Skelton, and K. Kreutz-Delgado, “Control synthesis for a class of light and agile robotic tensegird structures,” 2003 American Control Conference (ACC), vol. 6, pp. 5245–5251, 2003.

[24] A. S. Wolodsen, M. de Oliveira, and R. Skelton, “Modelling and control of non-minimal non-linear realisations of tensegird systems,” International Journal of Control, vol. 82, no. 3, pp. 389–407, mar 2009.

[25] R. Skelton and J. M. Mirats-Tur, “Nonlinear control of non-minimal tensegird models,” in Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, dec 2009, pp. 6662–6667.

[26] J. M. Mirats Tur and S. H. Juan, “Tensegird frameworks: Dynamic analysis review and open problems,” Mechanism and Machine Theory, vol. 44, no. 1, pp. 1–18, jan 2009.

[27] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer, “Flop and roll: Learning robust goal-directed locomotion for a Tensegird Robot,” in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, sep 2014, pp. 2236–2243.

[28] A. Iscen, K. Caluwaerts, J. Bruce, A. Agogino, V. SunSpiral, and
K. Tumer, “Learning Tensegrity Locomotion Using Open-Loop Control Signals and Coevolutionary Algorithms,” Artificial Life, vol. 21, no. 2, pp. 119–140, may 2015.

[29] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. Sun-Spiral, P. Abbeel, and S. Levine, “Deep reinforcement learning for tensegrity robot locomotion,” in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, may 2017, pp. 634–641.

[30] C. Sultan, M. Corless, and R. E. Skelton, “Symmetrical reconfiguration of tensegrity structures,” International Journal of Solids and Structures, vol. 39, no. 8, pp. 2215–2234, apr 2002.

[31] C. Sultan and R. Skelton, “Deployment of tensegrity structures,” International Journal of Solids and Structures, vol. 40, no. 18, pp. 4637–4657, 2003.

[32] K. Caluwaerts and J. P. Carbajal, “Energy conserving constant shape optimization of tensegrity structures,” International Journal of Solids and Structures, vol. 58, pp. 117–127, apr 2015.

[33] K. Worthmann, M. W. Mehrrez, M. Zanon, G. K. I. Mann, R. G. Gosine, and M. Diehl, “Model Predictive Control of Nonholonomic Mobile Robots Without Stabilizing Constraints and Costs,” IEEE Transactions on Control Systems Technology, vol. 24, no. 4, pp. 1394–1406, jul 2016.

[34] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. Tseng, “A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous vehicles,” 2007 46th IEEE Conference on Decision and Control, pp. 2980–2985, 2007.

[35] K. Caluwaerts, J. Despraz, A. Iscen, A. P. Sabelhaus, J. Bruce, B. Schrauwzen, and V. Sun-Spiral, “Design and control of compliant tensegrity robots through simulation and hardware validation,” Journal of the Royal Society Interface, vol. 11, no. 98, pp. 20140520–20140520, jul 2014.

[36] K. Caluwaerts, J. Bruce, J. M. Friesen, and V. Sun-Spiral, “State estimation for tensegrity robots,” in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, may 2016, pp. 1860–1865.

[37] K. J. Astrom and R. M. Murray, Feedback systems: an introduction for scientists and engineers. Princeton university press, 2008.

[38] H. J. Schek, “The force density method for form finding and computation of general networks,” Computer Methods in Applied Mechanics and Engineering, vol. 3, no. 1, pp. 115–134, jan 1974.

[39] A. Tibert and S. Pellegrino, “Review of Form-Finding Methods for Tensegrity Structures,” International Journal of Space Structures, vol. 18, no. 4, pp. 209–223, dec 2003.

[40] H. C. Tran and J. Lee, “Advanced form-finding of tensegrity structures,” Computers & Structures, vol. 88, no. 3–4, pp. 237–246, feb 2010.

[41] C. Calladine, “Buckminster Fuller’s Tensegrity structures and Clerk Maxwell’s rules for the construction of stiff frames,” International Journal of Solids and Structures, vol. 14, no. 2, pp. 161–172, jan 1978.

[42] R. Connelly and W. Whiteley, “The Stability of Tensegrity Frameworks,” International Journal of Space Structures, vol. 7, no. 2, pp. 153–163, 1992.

[43] S. Pellegrino and C. Calladine, “Matrix analysis of statically and kinematically indeterminate frameworks,” International Journal of Solids and Structures, vol. 22, no. 4, pp. 409–428, jan 1986.

[44] R. Motro, Tensegrity: Structural systems for the future. Butterworth-Heinemann, 2003.

[45] J. Lofberg, “YALMIP : a toolbox for modeling and optimization in MATLAB,” in 2004 IEEE International Conference on Robotics and Automation. IEEE, 2004, pp. 284–289.

[46] F. Borrelli, Constrained Optimal Control of Linear and Hybrid Systems, ser. Lecture Notes in Control and Information Sciences. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 290.

[47] “Gurobi Optimizer Reference Manual,” Gurobi Optimization Inc., 2016.

Andrew P. Sabelhaus received the B.S. degree in mechanical engineering from Villanova University, PA, USA in 2015 and his M.S. degree in mechanical engineering from the University of California Berkeley, CA, USA in 2017. He was a DoD SMART fellow at Berkeley, where his research involved uncertainty estimation and planning for tensegrity robot locomotion. He is currently a Research Scientist at the U.S. Army Research Lab, Vehicle Technology Directorate, Aberdeen, MD, USA.

Huajing Zhao received the B.E. degree in mechanical engineering and automatic manufacturing from Harbin Institute of Technology, China in 2016, and the M.Eng degree in mechanical engineering from the University of California Berkeley, USA in 2017. She is currently pursuing a M.S. degree in the department of robotics at the University of Michigan, Ann Arbor, MI, USA. She is a member of Automotive Research Center (ARC), and received Best Student Poster Finalist Award at the 2018 ARC Annual Program Review.

Edward L. Zhu received his B.S. degree in mechanical engineering from Stanford University, CA, USA in 1996, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA in 1999 and 2003 respectively. He is a research scientist at NASA Ames Research Center, Moffett Field, CA, where he has been since 2004. He has over 70 publications in the fields of machine learning, soft robotics, rocket analysis, multi-agent systems, reinforcement learning, evolutionary systems and visualization of complex systems. He has received awards for publications in both learning and in visualization.

Adrian K. Agogino received the B.S. degree in computer engineering from the University of California San Diego, La Jolla, CA, USA in 1996, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California Berkeley, CA, USA in 1999 and 2003 respectively. He is a research scientist at NASA Ames Research Center, Moffett Field, CA, where he has been since 2004. He is currently the Roscoe and Elizabeth Hughes Director of the Blum Center for Developing Economies.

Alice M. Agogino received her PhD in Management Science & Engineering from Stanford University, MS in Mechanical Engineering from UC Berkeley and BS in Mechanical Engineering from University of New Mexico. She is currently the Roscoe and Elizabeth Hughes Professor of Mechanical Engineering at UC Berkeley, and serves as Chair of the Graduate Group in Development Engineering as well as Education Director of the Blum Center for Developing Economies. Prof. Agogino has authored over 300 peer-reviewed publications, is a member of the National Academy of Engineering (NAE) and has won numerous teaching, mentoring, best paper, and research awards. She has supervised 53 PhD dissertations and 193 MS theses/reports.