Model-Predictive Control of a Flexible Spine Robot

Andrew P. Sabelhaus*[1†], *Student Member, IEEE*, Abishek K. Akella[2], Zeerek A. Ahmad[3], Vytas SunSpiral[4†]

***Abstract*— The Underactuated Lightweight Tensegrity Robotic Assistive Spine (ULTRA Spine) project is an ongoing effort to develop a flexible, actuated backbone for quadruped robots. In this work, model-predictive control is used to track a trajectory in the robot's state space, in simulation. This is the first work that tracks an arbitrary trajectory, in closed-loop, in the state space of a spine-like tensegrity robot. The state trajectory used here corresponds to a bending motion of the spine, with translations and rotations of the three moving vertebrae. The controller uses a linearized model of the system dynamics, computed at each timestep, and has both constraints and weighted penalties to reduce linearization errors. For this robot, which measures 26cm x 26cm x 45cm, the tracking errors converge to less than 0.5cm even with disturbances, indicating that the controller is stable and could be used on a physical robot in future work.**

I. INTRODUCTION

Quadruped (four-legged) robots are usually constructed with rigidity between their hips and shoulders, even when they are designed to solve difficult locomotion problems such as walking over uneven terrain [1], [2], [3], [4]. Quadrupeds with flexible bodies, particularly with actuation in a spine-like structure, often have single-degree-of-freedom mechanisms [5], [6], [7]. Actuated spines with many degrees-of-freedom could help with locomotion challenges, but few have been designed or implemented, partially due to the control challenges for such a structure. Even current actuated spines have only been controlled with kinematics-only models [8], [9], model-free control using central-pattern generators [6], [7], [10], or the replaying of open-loop inputs [11].

The Underactuated Lightweight Tensegrity Robotic Assistive Spine (ULTRA Spine) is a project to design and control a multi-degree-of-freedom flexible robotic spine for a quadruped robot [12]. This work presents closed-loop tracking controller for the ULTRA Spine. Fig. 1 shows the spine during a bending motion.

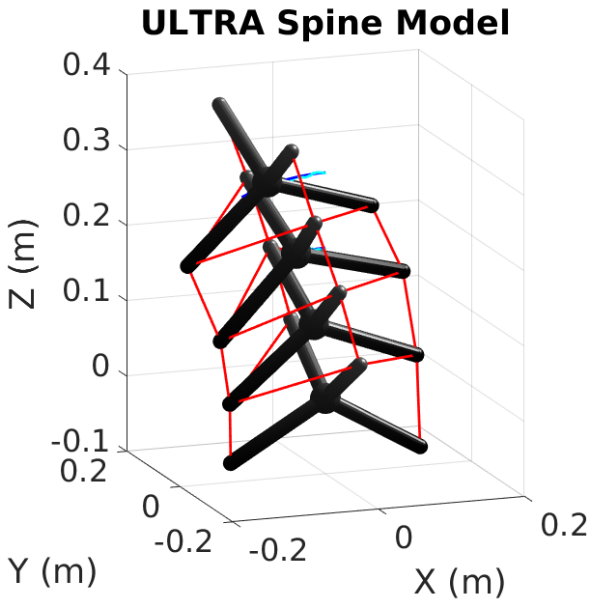

Fig. 1: Trajectory-tracking control for the flexible backbone robot (ULTRA Spine), mid-simulation, for a uniaxial bending trajectory. The rigid bodies (vertebrae) of the spine are in gray, cables in red, and the target trajectory for the top spine vertebra is in blue. This work uses a point-mass dynamics model, so the rigid vertebra bodies are for visualization only.

A. Tensegrity Robots and Control

The ULTRA Spine 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], and are inherently flexible. Many types of robots have been designed around a tensegrity system, which change their shape by adjusting the lengths of their cables so that they can roll [14], [15], [16], crawl [17], [18], [19], swim [20], [21], and climb [22], [23]. In particular, different models of tensegrity spines have been investigated [24], [25], [19], but the ULTRA Spine is one of the first uses of a tensegrity spine on a quadruped robot [12], [10].

Control of tensegrity robots has been limited to either analytical solutions for simple topologies [26], model-free techniques [17], [25], [19], or open-loop control [27], [28], [29], [22]. To the authors' knowledge, this is the first work to track trajectories in a tensegrity spine in the robot's state space, and one of the first to do so for any tensegrity robot.

B. Why Model-Predictive Control?

Prior unpublished attempts at controlling the ULTRA Spine with linear controllers (such as LQR) have not been stable. As in this work, only position states have been available as a reference trajectory, without velocity states and without a corresponding input trajectory. Combined with

* *corresponding author.*
†Authors with the NASA Ames Intelligent Robotics Group and the Dynamic Tensegrity Robotics Lab, Moffett Field CA 94035, USA
[1]A.P. Sabelhaus is with the Department of Mechanical Engineering, University of California Berkeley, USA apsabelhaus@berkeley.edu
[2]A.K. Akella is with Levant Power Corp., 475 Wildwood Ave, Woburn MA 01801, USA akakella@berkeley.edu
[3]Z.A. Ahmad is with Velo3D Inc., 1001 Belford Dr., San Jose, CA 95132, USA zeerekahmad@gmail.com
[4]V. SunSpiral is with SGT Inc., Greenbelt, MD 20770, USA, working at NASA Ames Research Center, Moffett Field CA 94035, USA vytas.sunspiral@nasa.gov

the hybrid system dynamics of the spine (due to cable slackness), these controller formulations have had significant linearization errors, and have exhibited large instabilities.

Model-Predictive Control (MPC) is a control technique that incorporates state and input constraints on a system [30]. MPC has been used for many constrained robotics control problems [31], [32], [33], [34]. This work uses the constraints in an MPC controller enforce small linearization errors and reduce cable slackness and stabilize the robot.

All software associated with this work is available under an open-source license online[1].

II. ULTRA SPINE SYSTEM DYNAMICS

This ULTRA Spine state-space model uses a set of rigid-body states for each vertebra (12 states per rigid body, 3 vertebrae, 36 states.). However, as in much of the literature [15], [35], [23], a point mass approximation is used for tractability. Each vertebra is modeled as five point masses (Fig. 2). The dynamics derivation below contains the translations and rotations that map the rigid body states onto point-mass positions. The dynamics are formulated using Lagrange's equations, then implemented and solved symbolically in MATLAB. Since an analytical dynamics model is needed for Model-Predictive Control (sect. IV), prior work in the field including kinematics models [22], [35], [23] and numerical methods [36], [15], [19] could not be used.

This controller uses a discrete-time dynamics model, but the dynamics derivation occurs in continuous time. The discretization is discussed alongside the linearization in sect. IV. We assume full knowledge of the system states at each timestep. The continuous-time system model is of the form

$$\begin{array}{rcll} \dot{\xi} & = & g(\xi, u) & \xi \in \mathbb{R}^{36} \\ y & = & \xi & u \in \mathbb{R}^{24} \end{array} \tag{1}$$

where ξ is the state vector. As shown in Fig. 1, there are 24 cables in this model, each treated as an input: 12 vertical cables extending vertically along the spine's edges, and 12 "saddle" cables holding the vertebrae apart. Note that since the spine vertebrae should never be rotated by more than even 90 degrees, the Euler angles in the rigid body states do not need to be explicitly constrained to lie between $[0, 2\pi]$.

A. Topology

The topology of this spine is defined by the connections between its cables and its rigid bodies (vertebrae.) Each vertebra consists of 5 points masses: one at its center, and one at each end of its four "legs." Each leg extends outward at a 30° angle relative to the horizontal axis, and is 15cm long. This puts the end of each leg at 7.5cm above or below the center node. Fig. 2 shows one vertebra, with its six rigid body states: Cartesian positions (X, Y, Z) and rotations (θ, γ, ϕ). The five point masses are 0.142kg each, for a total mass of 0.71kg per vertebra. These dimensions and masses correspond to an early hardware prototype of the robot [12].

[1]https://github.com/BerkeleyExpertSystemTechnologiesLab/ultra-spine-simulations/tree/acc2017

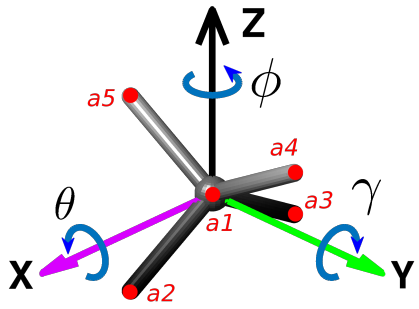

Fig. 2: A single spine vertebra with its local coordinate system. The point mass locations $a_1 ... a_5$ are labeled in red.

For a single vertebra, the locations of each of the four point masses at the end of a spine, with respect to the center of the vertebra, are, in centimeters,

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix} = \begin{bmatrix} 0 & 13 & -13 & 0 & 0 \\ 0 & 0 & 0 & 13 & -13 \\ 0 & -7.5 & -7.5 & 7.5 & 7.5 \end{bmatrix}$$

This notation uses a_1 as the location of the center of the vertebra, corresponding to the (X, Y, Z) states of each vertebra in the state vector.

The positions of each of the four point masses as the ends of the "legs" of a vertebra are calculated using rotation matrices corresponding to the three Euler angles of that vertebra, then offset by the position of the center node. For vertebra j, the Cartesian positions of the center node and Euler angles are picked out of the state vector:

$$\begin{bmatrix} x_j \\ y_j \\ z_j \\ \theta_j \\ \gamma_j \\ \phi_j \end{bmatrix} = \begin{bmatrix} \xi_{(j-1)\times 12+1} \\ \xi_{(j-1)\times 12+2} \\ \xi_{(j-1)\times 12+3} \\ \xi_{(j-1)\times 12+4} \\ \xi_{(j-1)\times 12+5} \\ \xi_{(j-1)\times 12+6} \end{bmatrix}, \quad j = 1...3$$

Then, denoting $R_\theta, R_\gamma, R_\phi$ as the 3D rotation matrices for each Euler angle, and with $\hat{\mathbf{e}}_\mathbf{x}, \hat{\mathbf{e}}_\mathbf{y}$ and $\hat{\mathbf{e}}_\mathbf{z}$, as the unit vectors in each Cartesian direction, the final position of each point mass for vertebra j is then

$$\begin{bmatrix} \mathbf{q}_{1j} \\ \mathbf{q}_{2j} \\ \mathbf{q}_{3j} \\ \mathbf{q}_{4j} \\ \mathbf{q}_{5j} \end{bmatrix} = \begin{bmatrix} x_j\hat{\mathbf{e}}_\mathbf{x} + y_j\hat{\mathbf{e}}_\mathbf{y} + z_j\hat{\mathbf{e}}_\mathbf{z} \\ R_\phi(\phi_j)R_\gamma(\gamma_j)R_\theta(\theta_j)a_2 + x_j\hat{\mathbf{e}}_\mathbf{x} + y_j\hat{\mathbf{e}}_\mathbf{y} + z_j\hat{\mathbf{e}}_\mathbf{z} \\ R_\phi(\phi_j)R_\gamma(\gamma_j)R_\theta(\theta_j)a_3 + x_j\hat{\mathbf{e}}_\mathbf{x} + y_j\hat{\mathbf{e}}_\mathbf{y} + z_j\hat{\mathbf{e}}_\mathbf{z} \\ R_\phi(\phi_j)R_\gamma(\gamma_j)R_\theta(\theta_j)a_4 + x_j\hat{\mathbf{e}}_\mathbf{x} + y_j\hat{\mathbf{e}}_\mathbf{y} + z_j\hat{\mathbf{e}}_\mathbf{z} \\ R_\phi(\phi_j)R_\gamma(\gamma_j)R_\theta(\theta_j)a_5 + x_j\hat{\mathbf{e}}_\mathbf{x} + y_j\hat{\mathbf{e}}_\mathbf{y} + z_j\hat{\mathbf{e}}_\mathbf{z} \end{bmatrix}$$

Thus, the positions of all 15 point masses have been defined in terms of the 18 position/angle states in the state vector. The other 18 states (their derivatives) are not used.

B. Dynamics Model

The distances between cable-connected nodes are stored as vectors l_i for $i = 1...24$ cables, each as a function of the system states ξ. The cables are modeled as spring-damper systems, where the (scalar) tension force on cable i is:

$$F_i = k(||l_i|| - p_i) - c||\dot{l_i}|| \qquad i = 1...24 \tag{2}$$

where $k = 2000\frac{N}{m}$ is the spring constant, $c = 100\frac{Ns}{m}$ is the damping constant, $||l_i||$ is the scalar length of the cable, and p_i is the rest length of cable i. This model assumes that the rest lengths of the cables are the inputs to the system, and that they can be controlled directly:

$$u = [\quad p_1 \quad p_2 \quad ... \quad p_{24} \quad]^\top \qquad u \in \mathbb{R}^{24}$$

The 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: the cable tensions are calculated first given the equations above, then those tension values are rectified if they are negative:

$$F_i = (F_i > 0) \times F_i \tag{3}$$

where the parenthetical expression evaluates to a boolean 1 or 0 in MATLAB. Third, these values are plugged in for F_i below in the solved dynamics.

Finally, Lagrange's equations are used to calculate the system dynamics in continuous time. For all $j = 1...3$ vertebra with $k = 1...5$ point masses at $m = 0.142kg$ each, and where $q_{(z)kj}$ is the Cartesian coordinate in the z-direction for point mass $\mathbf{q}_{kj}$, and where ξ_d is the d-th element of the state vector,

$$T = \frac{1}{2}\sum_{j=1}^{3}\sum_{k=1}^{5} m\|\dot{\mathbf{q}}_{kj}\|_2^2 \tag{4}$$

$$V = \sum_{j=1}^{3}\sum_{k=1}^{5} mgq_{(z)kj} \tag{5}$$

$$L = T - V \tag{6}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\xi}_d} - \frac{\partial L}{\partial \xi_d} = \sum_{i=1}^{24}\frac{\partial F_i}{\partial \xi_d}, \quad d = \{1..6, 13..18, 25..30\} \tag{7}$$

Here, (7) contains 18 equations: one for each of the position and angle states, for each of the three vertebrae. Note that the forces from eqn. (2-3) are not included in eqn. (5), they are instead in the right-hand side of eqn. (7).

The right-hand side and left-hand side of equation 7 are then solved symbolically, then equated, and the derivatives of each state were solved for (since each $\mathbf{q}_{kj}$ is expressed in terms of ξ.) This was performed using a combination of the MATLAB commands `solve`, `simplify`, and `parfeval` for multi-threading. The resulting equations for $g(\xi, u)$ are very long, so are not provided here, but are available online[2].

[2]https://github.com/BerkeleyExpertSystemTechnologiesLab/ultra-spine-simulations/tree/master/dynamics/3d-dynamics-symbolicsolver

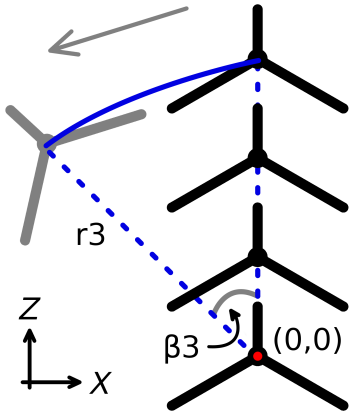

Fig. 3: Bending trajectory for the top vertebra of the spine in the X-Z plane, exaggerated for visibility. The vertebrae rotate counterclockwise around the origin at a constant radius r_j, swept out by angle β_j. Solid blue line shows the position of the center of the vertebra. The middle two vertebrae rotate in similar manners. The lowest vertebra is fixed, and is not part of the dynamics.

III. CONTROL OBJECTIVES

In this work, the ULTRA Spine robot is controlled in a bending motion. This trajectory was motivated by prior work [12], where forward kinematics for this spine were used to determine the vertebra positions when one set of its vertical cables were retracted. The trajectory had $n = 80$ timesteps.

No a-priori dynamic trajectories were available for this model: no method existed for creating a dynamic trajectory from a kinematic one. Consequently, the controller in sect. IV does not include tracking of control inputs or velocities.

A. Kinematic Trajectory

The spine starts in an upright position, then rotates counterclockwise around the -Y axis by an angle β_j for vertebra j, staying in the 2-dimensional X-Z plane (Fig. 3). This "bending" trajectory includes translations in X and Z as well as rotations in γ for each vertebra.

As per sect. II-A, the spine vertebrae are separated by 10cm vertically in their starting position:

$$[z_1(0) \quad z_2(0) \quad z_3(0)] = [0.1 \quad 0.2 \quad 0.3]$$

These initial heights are the radius of the rotation: $r_j = z_j(0)$. Consequently, the reference positions of each vertebra over time, x_j^{ref} and z_j^{ref}, are:

$$x_j^{ref} = r_j \sin(\beta_j), \quad z_j^{ref} = r_j \cos(\beta_j)$$

The rotations of each vertebra around its inertial frame, in γ, were the same as the sweep angle for that vertebra. This kept the vertebrae parallel to the radius of rotation:

$$\gamma_j^{ref} = \beta_j \tag{8}$$

Each vertebra had a different maximum sweep angle, to be consistent with the kinematics simulations in prior work [12]. These angles were:

$$[\beta_{1max}, \quad \beta_{2max}, \quad \beta_{3max}] = \left[\frac{\pi}{16}, \quad \frac{\pi}{12}, \quad \frac{\pi}{8}\right]$$

The range from 0 to $\beta_{j(max)}$ was divided linearly between each of the $n = 80$ timesteps. All other reference trajectory states in ξ^{ref} were zero.

IV. CONTROLLER FORMULATION

A model-predictive controller was used to track the above trajectory ξ_{ref}. At each timestep t of the controller, the following constrained finite-time optimal control problem (CFTOC) was solved, generating the sequence of control inputs $U_{t\to t+N|t} = \{u_{t|t}, ..., u_{t+N|t}\}$, with a window of $N = 10$. The notation $t + k|t$ represents a value at the timestep $t + k$, as given or predicted at timestep t ([37], Ch. 4.) Then, the first input $u_{t|t}$ is applied to the system, and the simulation advances to timestep $t+1$, and the problem repeats. Note that no terminal costs or constraints are included here, and thus stability can only be experimentally concluded, not proven.

$$\min_{U_{t\to t+N|t}} \sum_{k=0}^{N} J(\xi_{t+k|t}, u_{t+k|t}, \xi^{ref}_{t+k}) \tag{9}$$

subj. to:

$$\xi_{t+k+1} = A_t\xi_{t+k} + B_t u_{t+k} + c_t \tag{10}$$
$$u_{min} \le u_{t+k} \le u_{max} \tag{11}$$
$$\|u_t - u_{(t-1)}\|_\infty \le w_1 \tag{12}$$
$$\|u_{t+k} - u_t\|_\infty \le w_2, \ k = 1..(N-1) \tag{13}$$
$$\|u_{t+N} - u_t\|_\infty \le w_3 \tag{14}$$
$$\|\xi(1:6)_{t+k} - \xi(1:6)_{t+k-1}\|_\infty \le w_4 \tag{15}$$
$$\|\xi(13:18)_{t+k} - \xi(13:18)_{t+k-1}\|_\infty \le w_5 \tag{16}$$
$$\|\xi(25:30)_{t+k} - \xi(25:30)_{t+k-1}\|_\infty \le w_6 \tag{17}$$
$$\xi(3)_{t+k} + w_7 \le \xi(15)_{t+k} \tag{18}$$
$$\xi(15)_{t+k} + w_7 \le \xi(27)_{t+k} \tag{19}$$

Here, $N = 10$ is the horizon length (a scalar), $w_1...w_7$ are constant scalar weights, and $\xi(i)_{t+k}$ denotes the i-th element of the state vector at time $t+k$ as predicted at time t. The dynamics constraint, (10), consists of the linearized and discretized system at time t, calculated as

$$A_t = \frac{\partial g(\xi,u)}{\partial \xi}\Big|_{\substack{\xi=\xi_t \\ u=u_{t-1}}} \qquad B_t = \frac{\partial g(\xi,u)}{\partial u}\Big|_{\substack{\xi=\xi_t \\ u=u_{t-1}}}$$
$$c_t = g(\xi_t, u_{(t-1)}) - A_t\xi_t - B_t u_{(t-1)}$$

The discretization occurs as A_t, B_t, and c_t are calculated, via a simple finite-difference Euler discretization, with $k = 0.001$, the same as the timestep for t. At each timestep of the simulation, these matrices are calculated by numerically differentiating the equations of motion from eqn. (7) in MATLAB: the dynamics are forward simulated in each direction, and a finite-difference approximation is taken. This approach was chosen due to computational issues with calculating additional analytical derivatives of the dynamics.

This linearization was calculated at each timestep t and used for the optimization over the entire horizon, thus the notation A_t, B_t, c_t. Since no trajectory of inputs was available, linearizations used the prior state's input u_{t-1} instead. For the start of the simulation, $u_0 = \mathbf{0}$ was used. Note that since these linearizations are not at equilibrium points, the linear system is affine, with c_t being a constant vector offset.

The remaining constraints used have the following interpretations. Constraint (11) 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 I). This is the constraint that helps prevent the system from operating in the slack-cable regime, thus keeping it in one set of continous dynamics instead of behaving as a hybrid system. Constraints (12), (13), and (14) are smoothing terms on the inputs, which help with the lack of an input reference trajectory. Here $u_{(t-1)}$ is the most recent input at the start of the CFTOC problem. Constraints (15), (16), and (17) are smoothing terms on the states, limiting the deviation between successive states in the trajectory. These are needed to reduce linearization error, and are split so that the positions and angles of each vertebra could be weighted differently. Note from (15-17) that no velocity terms are constrained. Finally, noting that states $\xi(3)$, $\xi(15)$, and $\xi(27)$ are the z-positions of each vertebra as per sect. II-A, constraints (18) and (19) prevent the collision between adjacent vertebrae.

The cost function J, written with arbitrary time index j,

$$\begin{aligned} J(\xi_j, u_j, \xi^{ref}_j) = \\ &(\xi_j - \xi^{ref}_j)^\top Q^j (\xi_j - \xi^{ref}_j) + \\ &(\xi_j - \xi_{(j-1)})^\top S^j (\xi_j - \xi_{(j-1)}) + \\ &w_8 \|(u_j - u_{(j-1)})\|_\infty \end{aligned} \tag{20}$$

As before, w_8 is a scalar, while Q and S are constant diagonal weighting matrices. Here, Q penalizes the tracking error in the states, S penalizes the deviation in the states at one timestep to the next, and w_8 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. Nonzeros are at states $\xi_1...\xi_6$, $\xi_{13}...\xi_{18}$, and $\xi_{25}...\xi_{30}$, recalling that $\xi \in \mathbb{R}^{36}$. Raising each diagonal element to the power j puts a heavier penality on terms farther away on the horizon. These are defined as:

$$Q_j = diag(w_9,\ w_9,\ w_9 \mid w_{10},\ w_{10},\ w_{10} \mid 0...0) \in \mathbb{R}^{12\times12}$$
$$S_j = diag(w_{11},\ w_{11},\ w_{11} \mid w_{11},\ w_{11},\ w_{11} \mid 0...0) \in \mathbb{R}^{12\times12}$$
$$Q = diag(Q_1,\ Q_2,\ Q_3), \quad S = diag(S_1,\ S_2,\ S_3)$$

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

TABLE I: Controller weights and constants.

Constant:	Value:	Interpretation:
u_{min}	0.0 meters (cable)	Min. Cable Length
u_{max}	0.20 meters (cable)	Max. Cable Length
w_1	0.01 meters (cable)	Input Smooth., Horiz. Start
w_2	0.01 meters (cable)	Input Smooth., Horiz. Middle
w_3	0.10 meters (cable)	Input Smooth., Horiz. End
w_4	0.02 meters, radians	State Smooth., Bottom Vert.
w_5	0.03 meters, radians	State Smooth., Mid. Vert.
w_6	0.04 meters, radians	State Smooth., Top Vertebra
w_7	0.02 meters (vert. pos.)	Vertebra Collision
w_8	1 no units	Input Smoothing
w_9	25 no units	State Tracking, Vertebra Pos.
w_{10}	30 no units	State Tracking, Vert. Angle
w_{11}	3 no units	Input Difference Penalty

V. SIMULATION SETUP

Two simulations are presented in this work, one for the controller without disturbances and one for the controller with disturbances. All simulation work used the YALMIP toolbox in MATLAB [38], with Gurobi as the solver [39].

The timestep used for the trajectory was $\Delta k = 0.001$ seconds, which was also the dt for the dynamics simulation. The dynamics were forward-simulated using the Runge-Kutta method.

A. Disturbances

For the simulation with disturbances, a weighted random variable was added to each state in the simulation after each timestep. This was a constant multiplied by a sample from $\mathcal{N}(0,1)$, with $w_{12}\mathcal{N}(0,1)$ for the position and angle states and $w_{13}\mathcal{N}(0,1)$ for the velocities. For positions, w_{12} was $0.5mm$ and $0.0005rad$, and for the velocities, w_{13} was $0.2\frac{cm}{sec}$ and $0.0002\frac{rad}{sec}$.

Note that the disturbances on the position states are very large in comparison to the distance travelled by the vertebrae between timesteps. The largest displacements between timesteps in the reference trajectory are 1.5mm, present in the top vertebra in the X-direction. Thus, the weight w_{12} is roughly 33% the size of this displacement, so $w_{12}\mathcal{N}(0,1)$ has a large magnitude in proportion to the changes in states due to the controller.

VI. RESULTS

The optimization problem (9)-(20) took 0.5-1 sec. to solve at each timestep, using the Gurobi solver.

The controller tracked the positions of the vertebrae with extremely low error, after an initial transient response. Fig. 4 shows the paths of the vertebrae in the X-Z plane as they sweep through their counterclockwise bending motion, including the reference trajectory (blue), the resulting trajectory with MPC controller and no disturbances (green), and the result of the controller with disturbances (magenta). Fig. 5 shows a zoomed-in view of the top vertebra, which had the largest tracking errors of the three vertebrae.

The tracking errors for each state, for each vertebra, for both simulations are shown in Fig. 6. In both simulations, an initial transient is observed in the X-position and γ-angle states. This is possibly due to a zero initial velocity of the vertebrae, requiring the spine to rapidly move at the start of its simulation to "catch up" with the trajectory. After that, all errors trend to zero, with the expected oscillations in the noisy simulation.

Additionally, the absolute value of the sum of the errors for the positions and angles are calculated in Fig. 7. The total sum of the errors trend toward zero, as expected from Fig. 6.

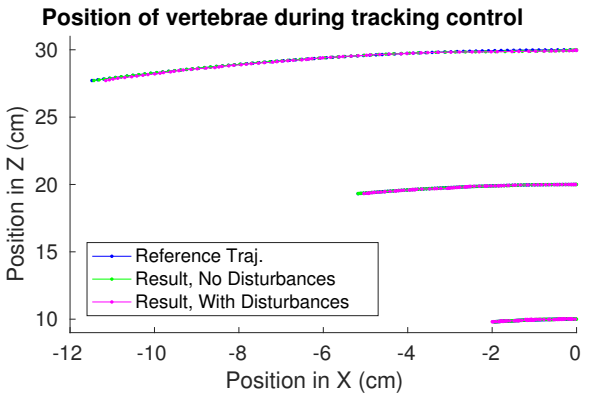

Fig. 4: Positions in the X-Z plane of all 3 of the vertebrae, including the reference and the two simulations (with/without disturbances), as the robot performs a counterclockwise bend. Blue trajectories are same as those in Fig. 3.

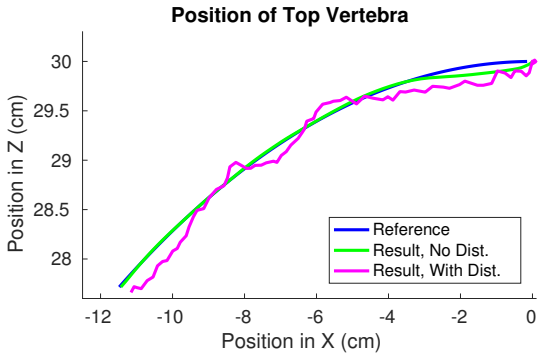

Fig. 5: Positions in the X-Z plane of the top vertebra, including the reference and the two simulations (with/without disturbances), as the robot performs a counterclockwise bend. The vertebra tracks the trajectory closely.

VII. DISCUSSION

The length of time taken to solve the optimization problem for the controller (0.5-1 sec.) was longer than the timestep of the simulation (0.001 sec.). Thus, the optimization procedure will need to be made more efficient before using this controller in hardware. This motivates the future use of two different approaches to reduce solver time: first, a symbolic Jacobian could be calculated for the A_t and B_t matrices, reducing the computation load in the linearization. More significantly, nonlinear model-predictive control (NMPC) could be used to remove the linearization entirely. These will be explored in future work.

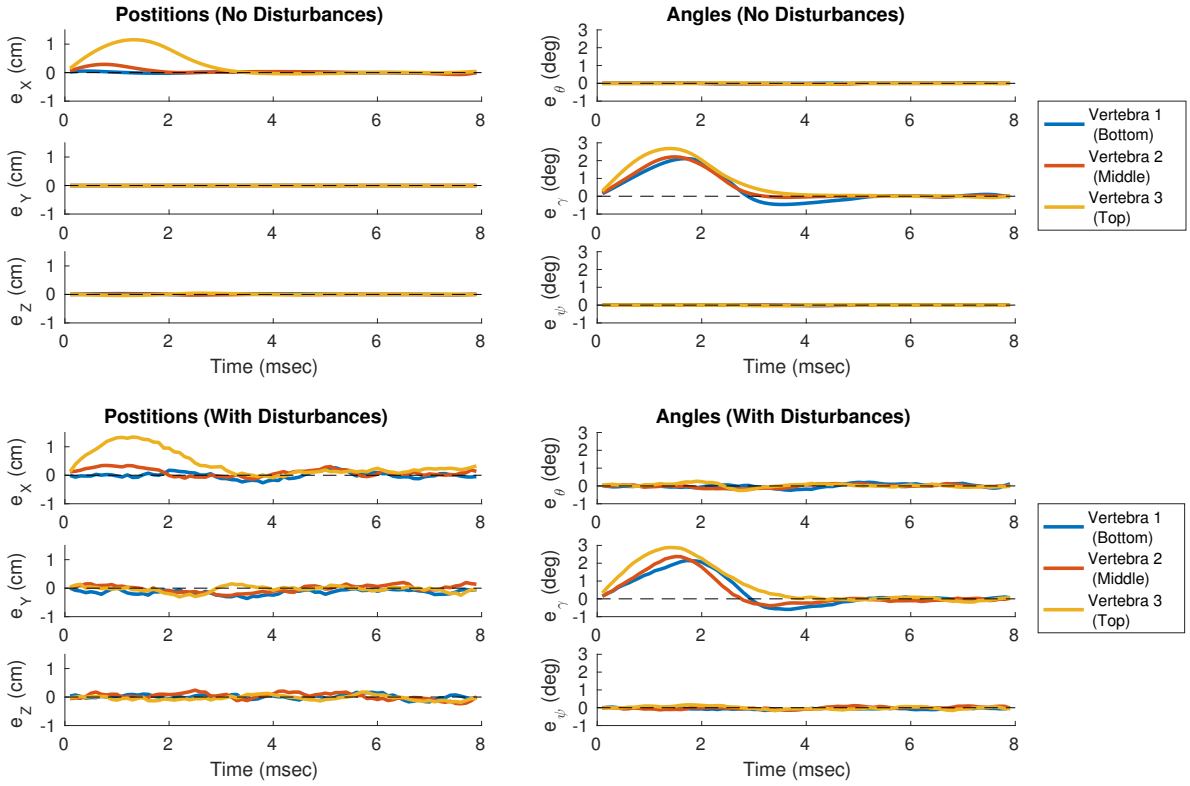

Fig. 6: Tracking errors for all three vertebrae, with and without disturbances. Cartesian position states (x, y, z) on the left, Euler angles (θ, γ, ψ) on the right. Position errors are in cm, rotation errors are in degrees.

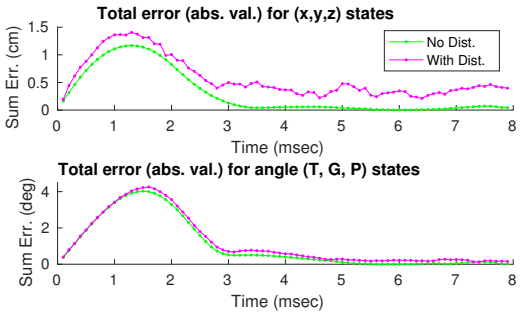

Fig. 7: Total tracking error (in absolute value), summing over all three vertebrae, for the position (XYZ, above) and angle $(\theta, \gamma, \phi$, below) states.

The many smoothing constraints and objective-function additions require that designers tune this controller for different use cases. Future work will look to reducing the need for these extra terms. Nonlinear MPC (NMPC) may help, since the dynamics linearization causes the problems that these terms address. Additionally, including a trajectory of inputs, instead of penalizing the magnitude of the inputs trajectory, will allow the controller to find a more realistic optimum.

This is the first work (to the authors' knowledge) that tracks a state-space trajectory of a tensegrity spine robot in closed-loop. The controller exhibits experimentally-stable behavior. The controller's low errors are promising for the use of this controller in hardware, given some computational improvements.

ACKNOWLEDGEMENTS

Many thanks to J. Friesen of [22], [23] for MATLAB code that contributed to the dynamics solutions in sect. II. Thanks to Profs. F. Borrelli and A. Packard who provided valuable guidance during the early stages of this project, during the course Experiential Advanced Control Design (ME C231A) at UC Berkeley. Thanks to Kyunam Kim of [15] for proofreading of sect. II. And many thanks to Prof. Alice Agogino of the Berkeley Emergent Space Tensegrities (BEST) Lab, as well as all the other BEST Lab members.

This research was supported by NASA Space Technology Research Fellowship no. NNX15AQ55H.

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*, mar 2003.

[2] L. R. Palmer and D. E. Orin, "Quadrupedal running at high speed over uneven terrain," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2007.

[3] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "BigDog, the Rough-Terrain Quadruped Robot," *IFAC Proceedings Volumes*, 2008.

[4] 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.

[5] M. Khoramshahi, A. Sprowitz, A. Tuleu, M. N. Ahmadabadi, and A. J. Ijspeert, "Benefits of an active spine supported bounding locomotion with a small compliant quadruped robot," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, may 2013.

[6] K. Weinmeister, P. Eckert, H. Witte, and A.-J. Ijspeert, "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.

[7] P. Eckert, A. Sprowitz, H. Witte, and A. J. Ijspeert, "Comparing the effect of different spine and leg designs for a small bounding quadruped robot," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2015.

[8] 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.

[9] T. Horvat, K. Karakasiliotis, K. Melo, L. Fleury, R. Thandiackal, and A. J. Ijspeert, "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.

[10] 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.

[11] 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.

[12] 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 Conference (IDETC) Volume 5A: 39th Mechanisms and Robotics Conference*. ASME, aug 2015.

[13] R. Skelton, R. Adhikari, J.-P. Pinaud, Waileung Chan, and J. Helton, "An introduction to the mechanics of tensegrity structures," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, IEEE. IEEE, 2001.

[14] A. P. Sabelhaus, J. Bruce, K. Caluwaerts, P. Manovi, R. F. Firoozi, S. Dobi, A. M. Agogino, and V. SunSpiral, "System Design and Locomotion of SUPERball, an Untethered Tensegrity Robot," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2015.

[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.

[16] L.-h. Chen, K. Kim, E. Tang, K. Li, R. House, E. Jung, A. M. Agogino, A. Agogino, and V. SunSpiral, "Soft Spherical Tensegrity Robot Design Using Rod-Centered Actuation and Control," in *ASME International Design Engineering Technical Conference (IDETC) Mechanisms and Robotics Conference*. Charlotte, NC: American Society of Mechanical Engineers, 2016.

[17] C. Paul, F. Valero-Cuevas, and H. Lipson, "Design and control of tensegrity robots for locomotion," *IEEE Transactions on Robotics*, oct 2006.

[18] 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.

[19] 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*, dec 2015.

[20] K. W. Moored, S. A. Taylor, T. K. Bliss, and H. Bart-Smith, "Optimization of a tensegrity wing for biomimetic applications," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006.

[21] T. Bliss, T. Iwasaki, and H. Bart-Smith, "Central Pattern Generator Control of a Tensegrity Swimmer," *IEEE/ASME Transactions on Mechatronics*, apr 2013.

[22] J. Friesen, A. Pogue, T. Bewley, M. de Oliveira, R. Skelton, and V. SunSpiral, "DuCTT: A tensegrity robot for exploring duct systems," in *ICRA*, may 2014.

[23] J. M. Friesen, P. Glick, M. Fanton, P. Manovi, A. Xydes, T. Bewley, and V. Sunspiral, "The second generation prototype of a Duct Climbing Tensegrity robot, DuCTTv2," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2016.

[24] B. Mirletz, I.-W. Park, T. E. Flemons, A. K. Agogino, R. D. Quinn, and V. SunSpiral, "Design and Control of Modular Spine-Like Tensegrity Structures," in *The 6th World Conference of the International Association for Structural Control and Monitoring (6WCSCM)*, 2014.

[25] B. T. Mirletz, I.-W. Park, R. D. Quinn, and V. SunSpiral, "Towards bridging the reality gap between tensegrity simulation and robotic hardware," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2015.

[26] R. E. Skelton and M. C. de Oliveira, *Tensegrity systems*. Springer, 2009.

[27] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer, "Learning to Control Complex Tensegrity Robots," in *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, 2013.

[28] ——, "Flop and Roll: Learning Robust Goal-Directed Locomotion for a Tensegrity Robot," in *Proceedings of The 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, 2014.

[29] 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*, may 2015.

[30] D. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, 1990.

[31] K. Worthmann, M. W. Mehrez, 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*, jul 2016.

[32] Dongbing Gu and Huosheng Hu, "Receding horizon tracking control of wheeled mobile robots," *IEEE Transactions on Control Systems Technology*, jul 2006.

[33] 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 systems," *2007 46th IEEE Conference on Decision and Control*, 2007.

[34] P. B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, 2006.

[35] K. Caluwaerts, J. Bruce, J. M. Friesen, and V. SunSpiral, "State estimation for tensegrity robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2016.

[36] K. Caluwaerts, J. Despraz, A. Iscen, A. P. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of The Royal Society Interface*, jul 2014.

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

[38] J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. IEEE, 2004.

[39] "Gurobi Optimizer Reference Manual," Gurobi Optimization Inc., 2016.