Model and Data Based Approaches to the Control of Tensegrity Robots

Ran Wang, Raman Goyal, Suman Chakravorty, Robert E. Skelton

***Abstract*— This paper proposes two approaches to control the shape of the structure or the position of the end effector for a soft-robotic application. The first approach is a model-based approach where the non-linear dynamics of the tensegrity system is used to regulate position, velocity and acceleration to the specified reference trajectory. The formulation uses state feedback to obtain the solution for the control (tension in the strings) as a *linear programming* problem. The other model-free approach is a novel decoupled data-based control (D2C) which first optimizes a deterministic open-loop trajectory using a black-box (no actual model) simulation model and then develops a linear quadratic regulator around the linearized open-loop trajectory. A two-dimensional tensegrity robotic reacher is used to compare the results for both the approaches for a given cost function. The D2C approach is also used to study two more complex tensegrity examples whose dynamics is difficult to model analytically.**

I. Introduction

The design of high DOF soft robotic systems has attracted increasing interest in recent years [1], [2]. In this regard, tensegrity structures offer a tantalizing prospect for the principled design of such soft robotic systems. In this paper, we study the modeling and control of high DOF tensegrity robotic systems. In particular, we develop full and reduced order models of class-k tensegrity structures and study both model based and data based approaches to the control of such systems. We also delineate the relative advantages and disadvantages of these approaches.

A. Related Work

Tensegrity structures are designed by placing bars and strings in a methodical arrangement to yield certain desired properties [3]. The dynamics of a tensegrity system can be very accurately modeled as all the members in the system are 1-dimensional elements which only take uni-directional loading [4]. The absence of bending moments on any individual element not only allows for the accurate modeling of the system but also provides the minimum mass solution to various kinds of loading conditions in engineering mechanics [3], [5]. For the same shape, the stiffness of the structure can also be changed by changing the pre-tension in the strings. The minimal mass architecture along with the variable stiffness characteristic makes it suitable for soft robotic applications like planetary landers [6], flexible robots [7], [8], and deployable space structures [9]. Some of the researchers used model-based approach [7], [8] and some used learning/evolutionary algorithm based approach [6], [10], [11] to control the tensegrity structures but no discussion has been given in the past to compare the two methods.

The control of tensegrity systems amounts to the design of a nonlinear stochastic controller for a very high DOF complex nonlinear system. Controlling an unknown dynamical system adaptively has a rich history in control literature [12] [13]. This classical literature provides rigorous analysis about the asymptotic performance and stability of the closed-loop system, mostly for linear systems or finite state and control space systems. The optimal control of a possibly unknown nonlinear dynamical system with continuous state and action space is a significantly more challenging problem. Even with a known model, computing an optimal control law requires solving a dynamic programming problem. The 'curse of dimensionality' associated with dynamic programming makes solving such problems computationally intractable, except under special structural assumptions on the underlying system. *Learning to control* problems where the model of the system is unknown, or is too large or complex for a tractable control synthesis, also suffer from this computational complexity issues, in addition to the usual identifiability problems of adaptive control. For the model-based methods, the computational time is often negligible if analytical model is known.

The past several years have seen significant progress in deep neural networks based reinforcement learning approaches for controlling unknown dynamical systems, with applications in many areas like playing games [14], locomotion [15] and robotic hand manipulation [16]. A number of new algorithms that show promising performance are proposed [17] [18] [19] and various improvements and innovations have been continuously developed. However, despite excellent performance on a number of tasks, reinforcement learning (RL) is still considered very data intensive. The training time for such algorithms is typically really large. Moreover, high variance and reproducibility issues on the performance are also reported [20].

In recent work [21], we proposed a novel decoupled data based control (D2C) algorithm for learning to control an unknown nonlinear dynamical system. Our approach introduced a rigorous decoupling of the open-loop (planning) problem from the closed-loop (feedback control) problem. This decoupling allows us to come up with a highly sample efficient approach to solve the problem in a completely data based fashion. Our approach proceeds in two steps: (i) first, we optimize the nominal open-loop trajectory of the system using a blackbox simulation model, (ii) then we

*The codes can be found in our GitHub repository here

*R. Wang, R. Goyal, S. Chakravorty and R. E. Skelton are with the Department of Aerospace Engineering, Texas A&M University, Texas, USA. {rwang0417, ramaniitrgoyal92, schakrav, bobskelton}@tamu.edu

identify the linear system governing perturbations from the nominal trajectory using random input-output perturbation data, and design an LQR controller for this linearized system. We have shown that the performance of D2C algorithm is approximately optimal, in the sense that the decoupled design is near optimal to second order in a suitably defined noise parameter [21]. In this paper, we show the application of the D2C data based control approach to complex and high DOF tensegrity robotic systems, including challenging cases of hard-to-model soft contact constraints and dynamic fluid interactions.

The contributions of the paper are as follows: we show how to model general tensegrity systems and find reduced order models for control design. We present a model based control approach and the application of the data based D2C control approach to such tensegrity systems, and present the relative advantages and disadvantages of the respective methods. We hope the comparison will help to determine the suitable applications for both methods and construct a systematic control solution for tensegrity soft robots.

II. Dynamics of General Tensegrity Structures

This section provides a brief overview of the non-linear dynamic model of a general tensegrity structure. The final form of class-1 tensegrity dynamics is formulated as a second-order matrix differential equation [4].

$$\ddot{N}M_s + NK_s = W, \tag{1}$$

$$M_s = \begin{bmatrix} C_{nb}(C_b^\mathsf{T}\hat{J}C_b + C_r^\mathsf{T}\hat{m}_b C_r) & C_{ns}^\mathsf{T}\hat{m}_s \end{bmatrix}, \tag{2}$$

$$K_s = \begin{bmatrix} C_s^\mathsf{T}\hat{\gamma}C_{sb} - C_{nb}C_b^\mathsf{T}\hat{\lambda}C_b & C_s^\mathsf{T}\hat{\gamma}C_{ss} \end{bmatrix}, \tag{3}$$

where λ represents the force density (compressive force per unit length) in the bar, given by:

$$\hat{\lambda} = -\hat{J}\hat{l}^{-2}\lfloor \dot{B}^\mathsf{T}\dot{B} \rfloor - \frac{1}{2}\hat{l}^{-2}\lfloor B^\mathsf{T}(W - S\hat{\gamma}C_s)C_{nb}^\mathsf{T}C_b^\mathsf{T} \rfloor, \tag{4}$$

and $N = \begin{bmatrix} n_1 & n_2 & \cdots & n_{2\beta+\sigma} \end{bmatrix} \in \mathbb{R}^{3\times(2\beta+\sigma)}$ represents the matrix containing the node position vectors $n_i \in \mathbb{R}^{3\times 1}$, β represents the number of bars, and σ represents the number of string-to-string nodes. The acceleration vector corresponding to the i^{th} node is represented by $\ddot{n}_i$, which forms the matrix $\ddot{N} = \begin{bmatrix} \ddot{n}_1 & \ddot{n}_2 & \cdots & \ddot{n}_{2\beta+\sigma} \end{bmatrix}$. The bar matrix $B = \begin{bmatrix} b_1 & b_2 & \cdots & b_\beta \end{bmatrix} \in \mathbb{R}^{3\times\beta}$ and string matrix $S = \begin{bmatrix} s_1 & s_2 & \cdots & s_\alpha \end{bmatrix} \in \mathbb{R}^{3\times\alpha}$ contain bar vectors b_i and string vectors s_i, respectively. The diagonal matrices $\hat{m}_b$ and $\hat{m}_s$ are formed by arranging bar masses and string point masses along the diagonal elements, respectively, and $\hat{J}$ is a diagonal matrix with $J_i = \frac{m_{b_i}}{12} + \frac{m_{b_i} r_{b_i}^2}{l_i^2}$ as the diagonal element that allows to accommodate the inertia in the bars. The connectivity matrices $C_{nb}, C_b, C_r, C_{ns}, C_s, C_{sb}$, and C_{ss} define the connections between different nodes to form bar vector, string vectors and string-to-string node positions. The external force matrix $W = \begin{bmatrix} w_1 & w_2 & \cdots & w_{2\beta+\sigma} \end{bmatrix}$ represents the matrix containing external force vector w_i corresponding to each node position vector n_i. The term 'force density in strings' is denoted by γ_i to describe tensile force per unit length in the i^{th} string member and $\hat{\gamma}$ represents the diagonal matrix formed from γ_i as its diagonal elements. Equation (4) provides the analytical formula to calculate the diagonal matrix $\hat{\lambda}$ where $\lfloor \circ \rfloor$ operator sets every off-diagonal element of the square matrix to zero. The diagonal matrix $\hat{l}$ is formed by the length of the bar members where l_i denotes the length of the i^{th} bar. Please refer to [4] for the detailed derivation of this dynamics model.

Dynamics Model for Class-k Tensegrity Structure

The model presented above is formulated for class-1 tensegrity structures. Any class-k structure can be described as a special case of class-1 structures by adding constraints on k nodes to have same node positions. These constraints are written in the linear form as:

$$NP = D, \tag{5}$$

where matrix $P \in \mathbb{R}^{(2\beta+\sigma)\times c}$ and matrix $D \in \mathbb{R}^{3\times c}$ can be written from the observation to provide constraints and c is the number of added constraints [4]. The added constraints can be accommodated in the class-1 tensegrity dynamics by introducing Lagrange multiplier $\Omega \in \mathbb{R}^{3\times c}$ that constraints the motion to a certain space by adding Lagrange constraint forces of the form ΩP^T. The full-order dynamics model for class-k tensegrity structure is written as:

$$\ddot{N}M_s + NK_s = W + \Omega P^\mathsf{T}, \tag{6}$$

with a modification for force density in the bars as:

$$\hat{\lambda} = -\hat{J}\hat{l}^{-2}\lfloor \dot{B}^\mathsf{T}\dot{B} \rfloor - \frac{1}{2}\hat{l}^{-2}\lfloor B^\mathsf{T}(W + \Omega P^\mathsf{T} - S\hat{\gamma}C_s)C_{nb}^\mathsf{T}C_b^\mathsf{T} \rfloor \tag{7}$$

The degree of freedom for the entire structure has reduced because of the above mentioned constraints. This reduction in the number of allowed displacement dimension can be captured in a reduced dimensional space by reducing the order of the model. The following second order matrix differential equation provides the reduced-order dynamic model as:

$$\ddot{\eta}_2 M_2 + \eta_2 K_2 = \widetilde{W}, \tag{8}$$

where $M_2 = U_2^\mathsf{T} M_s U_2$, $K_2 = U_2^\mathsf{T} K_s U_2$, $\widetilde{W} = WU_2 - \eta_1 U_1^\mathsf{T} K_s U_2$, and the remaining variables are generated from the Singular Value Decomposition (SVD) of the full-column rank matrix P as:

$$NP = NU\Sigma V^\mathsf{T} = N[U_1\ U_2]\begin{bmatrix}\Sigma_1 \\ 0\end{bmatrix}\begin{bmatrix}V^\mathsf{T}\end{bmatrix} = D, \tag{9}$$

where we use $[\eta_1\ \ \eta_2] \triangleq NU$ to get:

$$\eta_1 = DV\Sigma_1^{-1},\ \ \dot{\eta}_1 = 0,\ \ \ddot{\eta}_1 = 0. \tag{10}$$

Notice that η_1 represents the constraint space in the transformed coordinates and η_2 represents the reduced order space where the motion is present. The Lagrange multiplier is calculated at each step to introduce constraint forces by solving the following algebraic equation:

$$\eta_1 U_1^\mathsf{T} K_s M_s^{-1} U_1 + \eta_2 U_2^\mathsf{T} K_s M_s^{-1} U_1 = W M_s^{-1} U_1 + \Omega V \Sigma_1^\mathsf{T} U_1^\mathsf{T} M_s^{-1} U_1. \tag{11}$$

Please refer [4] for the analytic solution of this linear algebra problem.

III. Control for Class-K Tensegrity Systems

An important step in writing the control of this non-linear dynamic system into a linear programming problem is to be able to write the force densities in the bar $\lambda = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_\beta]^\mathsf{T}$ in terms of the linear function of force densities in the strings $\gamma = [\gamma_1 \ \gamma_2 \ \cdots \ \gamma_\alpha]^\mathsf{T}$. Let us write the i^{th} diagonal element of the matrix $\hat{\lambda}$ from Eq. (7) as:

$$\lambda_i = -J_i l_i^{-2} e_i^\mathsf{T} \lfloor \dot{B}^\mathsf{T} \dot{B} \rfloor e_i - \frac{1}{2} l_i^{-2} e_i^\mathsf{T} \lfloor B^\mathsf{T} (W + \Omega P^\mathsf{T} - S\hat{\gamma} C_s) C_{nb}^\mathsf{T} C_b^\mathsf{T} \rfloor e_i, \quad (12)$$

which is written using the identity $\hat{x}y = \hat{y}x$, for x and y being the column vectors, and stacking all the scalars into a column vector as:

$$\lambda = \Lambda\gamma + \tau, \quad (13)$$

where

$$\Lambda = \begin{bmatrix} \Lambda_1^\mathsf{T} & \Lambda_2^\mathsf{T} & \cdots & \Lambda_\beta^\mathsf{T} \end{bmatrix}^\mathsf{T}, \ \tau = \begin{bmatrix} \tau_1^\mathsf{T} & \tau_2^\mathsf{T} & \cdots & \tau_\beta^\mathsf{T} \end{bmatrix}^\mathsf{T},$$
$$\tau_i = -J_i l_i^{-2} ||\dot{b}_i||^2 - \frac{1}{2} l_i^{-2} b_i^\mathsf{T} (W + \Omega P^\mathsf{T}) C_{nb}^\mathsf{T} C_b^\mathsf{T} e_i,$$
$$\Lambda_i = \frac{1}{2} l_i^{-2} b_i^\mathsf{T} \widehat{S(C_s C_{nb}^\mathsf{T} C_b^\mathsf{T} e_i)}, \text{ for } i = 1, 2 \ \cdots \ \beta.$$

A. Controller for Reduced-Order Dynamics Model

In this section, we write down the control algorithm to control the position of certain nodes in the structure. Let us define the position of those nodes by $Y = LNR$, where L is a matrix that defines the x, y or z coordinates of the node and R matrix defines which nodes to be controlled. $\bar{Y}$ defines the final desired location of the nodes that we want to move from Y to $\bar{Y}$. Therefore, the error in the positions at any time is written as:

$$E = LNR - \bar{Y} = L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}, \quad (14)$$

and the first and second derivatives of the error with respect to time is written as:

$$\dot{E} = L\dot{\eta}_2 U_2^\mathsf{T} R, \quad \ddot{E} = L\ddot{\eta}_2 U_2^\mathsf{T} R, \quad (15)$$

where $\dot{\eta}_1 = \ddot{\eta}_1 = 0$ was used from the dynamics model formulation. Now, a second-order differential equation in the error dynamics is used to move the nodes from the current position to the desired position by aptly choosing the control gain parameters matrices Ψ and Θ:

$$\ddot{E} + \dot{E}\Psi + E\Theta = 0, \quad (16)$$
$$L\ddot{\eta}_2 U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta = 0. \quad (17)$$

Further substituting for $\ddot{\eta}_2$ from Eq. (8), the following equation is obtained:

$$L(WU_2 - \eta_1 U_1^\mathsf{T} K_s U_2 - \eta_2 U_2^\mathsf{T} K_s U_2) M_2^{-1} U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta = 0. \quad (18)$$

Rearranging the above equation to collect all the known and unknown terms together, we get:

$$LWU_2 M_2^{-1} U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta = L(\eta_1 U_1^\mathsf{T} K_s U_2 + \eta_2 U_2^\mathsf{T} K_s U_2) M_2^{-1} U_2^\mathsf{T} R. \quad (19)$$

Let us define the known left side of the equation as:

$$\mathcal{C} \triangleq LWU_2 M_2^{-1} U_2^\mathsf{T} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta, \quad (20)$$

and write the right hand side of the Eq. (19) as:

$$L(\eta_1 U_1^\mathsf{T} K_s U_2 + \eta_2 U_2^\mathsf{T} K_s U_2) M_2^{-1} U_2^\mathsf{T} R = LNK_s \underbrace{U_2 M_2^{-1} U_2^\mathsf{T}}_{M_{sn}} R. \quad (21)$$

We now take the i^{th} column of the above matrix and substitute for $K_s = C_s^\mathsf{T} \hat{\gamma} C_s - C_{nb}^\mathsf{T} C_b^\mathsf{T} \hat{\lambda} C_b C_{nb}$ to obtain:

$$LNK_s M_{sn} Re_i = LNC_s^\mathsf{T} \hat{\gamma} C_s M_{sn} Re_i - LNC_{nb}^\mathsf{T} C_b^\mathsf{T} \hat{\lambda} C_b C_{nb} M_{sn} Re_i, \quad (22)$$

and using the identity $\hat{x}y = \hat{y}x$ for the right-hand side terms gives:

$$LNK_s M_{sn} Re_i = LNC_s^\mathsf{T} \widehat{(C_s M_{sn} Re_i)} \gamma - LNC_{nb}^\mathsf{T} C_b^\mathsf{T} \widehat{(C_b C_{nb} M_{sn} Re_i)} \lambda. \quad (23)$$

Substituting for λ in terms of γ from Eq. (13) gives:

$$LNK_s M_{sn} Re_i = -LNC_{nb}^\mathsf{T} C_b^\mathsf{T} \widehat{(C_b C_{nb} M_{sn} Re_i)} \tau + \left(LNC_s^\mathsf{T} \widehat{(C_s M_{sn} Re_i)} - LNC_{nb}^\mathsf{T} C_b^\mathsf{T} \widehat{(C_b C_{nb} M_{sn} Re_i)} \Lambda \right) \gamma.$$

Now, substituting it back to the vector equation of Eq. (19), and stacking up all these matrices on left and vectors on right, we get:

$$\begin{bmatrix} \Gamma_1^\mathsf{T} & \Gamma_2^\mathsf{T} & \cdots & \Gamma_{n_r}^\mathsf{T} \end{bmatrix}^\mathsf{T} \gamma = \begin{bmatrix} \mu_1^\mathsf{T} & \mu_2^\mathsf{T} & \cdots & \mu_{n_r}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \quad (24)$$

where

$$\Gamma_i = LNC_s^\mathsf{T} \widehat{(C_s M_{sn} Re_i)} - LNC_{nb}^\mathsf{T} C_b^\mathsf{T} \widehat{(C_b C_{nb} M_{sn} Re_i)} \Lambda,$$
$$\mu_i = \mathcal{C} e_i + LNC_{nb}^\mathsf{T} C_b^\mathsf{T} \widehat{(C_b C_{nb} M_{sn} Re_i)} \tau,$$
$$\mathcal{C} = LWM_{sn} R + L\dot{\eta}_2 U_2^\mathsf{T} R\Psi + [L(\eta_1 U_1^\mathsf{T} + \eta_2 U_2^\mathsf{T})R - \bar{Y}]\Theta,$$
$$M_{sn} = U_2 M_2^{-1} U_2^\mathsf{T}, \quad \text{for} \quad i = 1, 2 \ \cdots \ n_r.$$

B. Controlling the Velocity and Acceleration

For controlling the node positions, we write the error in position as $E_p = L_p N R_p - \bar{Y}_p$ where subscript p is used for the position and write the final linear equation for the force densities in the string in a compact form as (refer Eq. (24)):

$$\Gamma_p \gamma = \mu_p, \quad \gamma \geq 0. \tag{25}$$

For controlling the velocity, we define the error in velocity of certain nodes as:

$$E_v = L_v \dot{\eta}_2 U_2^\mathsf{T} R_v - \dot{\bar{Y}}_v, \quad \dot{E}_v + E_v \Psi_v = 0, \tag{26}$$

and use a first-order differential equation to derive the error in velocity to zero. Only first derivative of error E_v is required as the control variable, force density γ in the strings come out at the same level of time derivative. Following the same derivation as used in the previous subsections, we write the linear equation to control the nodal velocities as:

$$\Gamma_v \gamma = \mu_v, \quad \gamma \geq 0. \tag{27}$$

To control the acceleration of the nodes, the error is defined as:

$$E_a = L_a \ddot{\eta}_2 U_2^\mathsf{T} R_a - \ddot{\bar{Y}}_a, \tag{28}$$

which can be directly converted to a linear equation in control variable by equating it to zero as $E_a = 0$. Following the same procedure, we get the linear algebra equation to solve for control variable as:

$$\Gamma_a \gamma = \mu_a, \quad \gamma \geq 0. \tag{29}$$

Finally, combining the Eqs. (25), (27), and (29) allows to simultaneously control the position, velocity and acceleration of different nodes in the structure.

$$\begin{bmatrix} \Gamma_p^\mathsf{T} & \Gamma_v^\mathsf{T} & \Gamma_a^\mathsf{T} \end{bmatrix}^\mathsf{T} \gamma = \begin{bmatrix} \mu_p^\mathsf{T} & \mu_v^\mathsf{T} & \mu_a^\mathsf{T} \end{bmatrix}^\mathsf{T}, \quad \gamma \geq 0. \tag{30}$$

IV. Decoupled Data Based Control (D2C) Algorithm

In this section, we propose a data-based approach to solve the tensegrity shape control problem as an alternative to the model-based approach in the previous section. In particular, we apply the so-called decoupled data-based control (D2C) algorithm that we recently proposed [21]. The D2C algorithm is a highly data-efficient Reinforcement Learning (RL) method that has shown to be much superior to state of the art RL algorithms such as the Deep Deterministic Policy Gradient (DDPG) in terms of data efficiency and training stability while retaining similar or better performance. In the following, we give a very brief introduction to the D2C algorithm (please see [21][22] for the relevant details).

Let us reconsider the dynamics of the tensegrity system given in Eq. (8) and rewrite it in a discrete time, noise perturbed state space form as follows:

$$x_{t+1} = f(x_t) + g(x_t)(u_t + \epsilon w_t), \tag{31}$$

where x_t is the state, u_t is the control, w_t is a white noise perturbation to the system, and ϵ is a small parameter that modulates the noise in the system. Suppose now that we have a finite horizon objective function: $J(x_0) = E[\sum_{t=0}^{T-1} c(x_t, u_t) + \phi(x_T)]$, where $c(x, u)$ denotes a running incremental cost, $\phi(x)$ denotes a terminal cost function and $E[\cdot]$ is an expectation operator taken over the sample paths of the system. The objective of the control design is then to design a feedback policy $u_t(x_t)$ that minimizes the cost function above and is given by:

$$J^*(x_0) = \min_{u_t(.)} E[\sum_{t=0}^{T-1} c(x_t, u_t) + \phi(x_T)]. \tag{32}$$

The D2C algorithm then proposes a 3 step procedure to approximate the solution to the above problem.

First, a noiseless open-loop optimization problem is solved to find an optimal control sequence, $\bar{u}_t^*$, that generates the nominal state trajectory $\bar{x}_t^*$:

$$J^*(x_0) = \min_{u_t} (\sum_{t=0}^{T-1} c(x_t, u_t) + \phi(x_T)). \tag{33}$$

subject to the zero noise nominal dynamics: $x_{t+1} = f(x_t) + g(x_t)u_t$.

Second, the perturbed time varying linear system around the nominal given by $\delta x_{t+1} = A_t \delta x_t + B_t \delta u_t$ is identified in terms of the time varying system matrices A_t, B_t.

Third, an LQR controller for the above time varying linear system is designed whose time varying gain is given by K_t. Finally, the control applied to the system is given by $u_t(x_t) = \bar{u}_t^* + K_t \delta x_t$.

Near Optimality of D2C: Let the mean and variance of the true cost function be given by $\bar{J}^*$ and $Var[J]^*$ respectively. Then, it can be shown that the D2C procedure outlined above is near optimal to the second order in the small parameter ϵ in the sense that $|\bar{J} - \bar{J}^*|$ is $O(\epsilon^2)$ and moreover, $\sqrt{|Var[J] - Var[J]^*|} = O(\epsilon^2)$, where $\bar{J}$ and $Var[J]$ are the mean and variance of the cost corresponding to the D2C policy above (refer to [21] for the details).

The following subsections provide details for each of the above-mentioned steps of the D2C algorithm.

A. Open Loop Trajectory Optimization

A first-order gradient descent based algorithm is proposed here for solving the open-loop optimization problem given in Eq. (33), where the underlying dynamics model is used as a blackbox, and the gradients are found from a sequence of rollout data with input perturbed by Gaussian noise. Let us denote the initial guess of the control sequence as $U^{(0)} = \{\bar{u}_t^{(0)}\}_{t=0}^{T-1}$, and the corresponding state trajectory $\mathcal{X}^{(0)} = \{\bar{x}_t^{(0)}\}_{t=0}^{T}$. The control policy is updated iteratively via:

$$U^{(n+1)} = U^{(n)} - \alpha \nabla_U \bar{J}|_{(\mathcal{X}^{(n)}, U^{(n)})}, \tag{34}$$

where $U^{(n)} = \{\bar{u}_t^{(n)}\}_{t=0}^{T-1}$ denotes the nominal control sequence after n iterations. $\bar{u}_t^{(n)}$ is a vector of the control value for each control channel at step t and α is the step size parameter. $\bar{J}|_{(\mathcal{X}^{(n)}, U^{(n)})}$ is the expected episodic cost

under $U^{(n)}$ and the corresponding state trajectory $\mathcal{X}^{(n)}$. The gradient vector is defined as:

$$\nabla_U \bar{J}|_{(\mathcal{X}^{(n)},U^{(n)})} = \begin{pmatrix} \frac{\partial \bar{J}}{\partial u_1} & \frac{\partial \bar{J}}{\partial u_2} & \cdots & \frac{\partial \bar{J}}{\partial u_T} \end{pmatrix}|_{(\mathcal{X}^{(n)},U^{(n)})}, \tag{35}$$

which is the gradient of the expected episodic cost w.r.t the control sequence after n iterations. The following paragraph elaborates on how to estimate the above gradient.

Here we define a rollout to be an episode in the simulation that starts from the initial settings to the end of the horizon with a control sequence. For each iteration, multiple rollouts are conducted sequentially, and the gradient vector is updated iteratively after each rollout. The expected episodic cost for the $(n+1)^{\text{th}}$ iteration is calculated by simulating a noiseless rollout with the control sequence $U^{(n)}$. The gradient vector is then estimated in a sequential manner as:

$$\nabla_U \bar{J}|^{j+1}_{(\mathcal{X}^{(n)},U^{(n)})} = (1-\frac{1}{j})\nabla_U \bar{J}|^{j}_{(\mathcal{X}^{(n)},U^{(n)})} + \frac{1}{j\sigma_{\delta u}}(J|_{(\mathcal{X}^{j,(n)},U^{j,(n)})} - \bar{J}|_{(\mathcal{X}^{(n)},U^{(n)})})(U^{j,(n)} - U^{(n)}), \tag{36}$$

where j and n denote the j^{th} rollout of the $(n+1)^{\text{th}}$ iteration. $\bar{J}|_{(\mathcal{X}^{(n)},U^{(n)})}$ is the expected episodic cost of the $(n+1)^{\text{th}}$ iteration while $J|_{(\mathcal{X}^{j,(n)},U^{j,(n)})}$ denotes the cost of the j^{th} rollout under control sequence $U^{j,(n)}$ and the corresponding state trajectory $\mathcal{X}^{j,(n)}$. Note that $U^{j,(n)} = \{\bar{u}_t^{(n)} + \delta u_t^{j,(n)}\}_{t=0}^{T-1}$ where $\{\delta u_t^{j,(n)}\}_{t=0}^{T-1}$ is the zero-mean, i.i.d Gaussian noise added as perturbation to the control sequence $U^{(n)}$. Scalar $\sigma_{\delta u}$ is the variance of the input perturbation and $\nabla_U \bar{J}|^{j+1}_{(\mathcal{X}^{(n)},U^{(n)})}$ denotes the gradient vector after j rollouts. The total rollout number r in one iteration is decided by the convergence of the gradient vector. After r rollouts, the control sequence is updated by Eq. (34) in which $\nabla_U \bar{J}|_{(\mathcal{X}^{(n)},U^{(n)})}$ is estimated by $\nabla_U \bar{J}|^{r}_{(\mathcal{X}^{(n)},U^{(n)})}$ using this iterative method. Repeat this until the cost converges and the optimized nominal control sequence is $\{\bar{u}_t^*\}_{t=0}^{T-1} = \{\bar{u}_t^N\}_{t=0}^{T-1}$. The idea is basically averaging the gradient vector over all the rollouts in an iterative way.

B. Linear Time-Varying System Identification

Closed-loop control design in step 2 of D2C requires the knowledge of the linearized system parameters A_t and B_t for $0 \le t \le T-1$. Here we use the standard least square method to estimate these parameters from input-output experiment data.

First start from the perturbed linear system about the nominal trajectory and estimate the system parameters A_t and B_t from: $\delta x_{t+1} = \hat{A}_t \delta x_t + \hat{B}_t \delta u_t$, where $\delta x_t^{(n)}$ is the state perturbation vector and $\delta u_t^{(n)}$ is the control perturbation vector we feed to the system at step t, n^{th} simulation. All the perturbations are zero-mean, i.i.d, Gaussian noise with covariance matrix σI. σ is a $o(u)$ small value selected by the user. $\delta x_{t+1}^{(n)}$ denotes the deviation of the output state vector from the nominal state after propagating for one step.

Run N simulations for each step and collect the input-output data:

$$Y = [\hat{A}_t \mid \hat{B}_t]X, \tag{37}$$

and write out the components:

$$Y = \begin{bmatrix} \delta x_{t+1}^{(1)} & \delta x_{t+1}^{(2)} & \cdots & \delta x_{t+1}^{(N)} \end{bmatrix},$$
$$X = \begin{bmatrix} \delta x_t^{(1)} & \delta x_t^{(2)} & \cdots & \delta x_t^{(N)} \\ \delta u_t^{(1)} & \delta u_t^{(2)} & \cdots & \delta u_t^{(N)} \end{bmatrix}. \tag{38}$$

Finally, using the standard least square method, the linearized system parameters are estimated as:

$$[\hat{A}_t \mid \hat{B}_t] = YX^{\mathsf{T}}(XX^{\mathsf{T}})^{-1}. \tag{39}$$

C. Closed Loop Control Design

Given the estimated perturbed linear system, we design a finite horizon, discrete time LQR [23] along the trajectory for each time step to minimize the cost function $J = \delta x_T^T Q \delta x_T + \sum_{t=0}^{T-1}(\delta x_t^T Q \delta x_t + u_t^T R u_t + 2\delta x_t^T N u_t)$, subjects to $\delta x_{t+1} = \hat{A}_t \delta x_t + \hat{B}_t \delta u_t$, where $u_t = -K_t \delta x_t$. The feedback gains are calculated as $K_t = (R + B^T P_{t+1} B)^{-1}(B^T P_{t+1} A + N^T)$, where P_t is solved in a back propagation fashion from the Riccati equation: $P_{t-1} = A^T P_t A - (A^T P_t B + N)(R + B^T P_t B)^{-1}(B^T P_t A + N^T) + Q, P_T = Q, N = 0$. The closed-loop control policy is $u_t(x_t) = \bar{u}_t^* - K_t \delta x_t$, where δx_t is the state deviation vector from the nominal state at time step t.

D. D2C Algorithm: Summary

The Decoupled Data Based Control (D2C) Algorithm is summarized in Algorithm 1.

Algorithm 1: D2C Algorithm

1) Solve the deterministic open-loop optimization problem for optimal open-loop control sequence and state trajectory $(\{\bar{u}_t^*\}_{t=0}^{T-1}, \{\bar{x}_t^*\}_{t=0}^{T})$ using gradient descent (Section IV-A).
2) Linearize and identify the LTV system parameters $(\hat{A}_t, \hat{B}_t)$ via least square (Section IV-B).
3) Solve the Riccati equations for each step along the nominal trajectory for feedback gain $\{K_t\}_{t=0}^{T-1}$.
4) Apply the closed-loop control policy,
while $t < T$ **do**

$$\begin{aligned} u_t &= \bar{u}_t^* - K_t \delta x_t, \\ x_{t+1} &= f(x_t) + B_t(u_t + \epsilon w_t), \\ \delta x_{t+1} &= x_{t+1} - \bar{x}_{t+1}^* \end{aligned} \tag{40}$$

$t = t + 1$.
end while

V. Simulation Results

In this section, we present the application of both the model based control and the data based D2C approach on three tensegrity robotic models with control inputs as the tension in the strings.

A. Structure and Task

We simulated all three robotic models in the MuJoCo simulator [24]: Reacher, Arm and 6-link swimmer. Each of the systems and their tasks are defined as follows:

a) Reacher: The minimal mass solution to compressive loading is provided by T-bar structure [3]. Tensegrity D-bar structure has also been shown to require less mass than a simple continuum bar to take the same load with the added advantage of deployability which makes it suitable for large motion space robotic applications. A T2D1 tensegrity structure is made by combining T-bar and D-bar structures as shown in Fig. 1(a). The two-dimensional structure has 22 bars and 22 strings. The bars in the system are connected by hinge joints reducing the degrees-of-freedom to 14 and the system is controlled by controlling the tension in the 22 control channels (strings). The control task is to move the top tip (end effector) to the target position (red dot).

b) Arm: The arm model is composed of 10 T-bar elements, made from rigidly attaching a vertical bar with a horizontal bar. The horizontal bars are aligned in the direction of the arm length. The first vertical bar is fixed as the base of the model. The model has 18 dimension of states and 38 control channels. The control task is to move the tip to the target position (red dot).

c) 6-link Swimmer: The 6-link swimmer is composed of 6 T-bar elements. The T-bar element here has the same structure as the arm model. Compared with the arm model, the first bar for the swimmer is not fixed so it could swim in the environment. The fluid density for the environment is 3000 kg/m^3, three times the density of water. The model swims by swaying its body so the horizontal bars would interact with the fluid to generate a forward force. Note that the diameter of the vertical bars is very small (difficult to see in the figure) so that they won't generate too much resistance while swimming. The model has 16 state variables and 22 control channels. The control task is to swim to the target position (red dot).

The initial positions of the models are shown in Fig. 1. The orange objects are the bars, and the grey ones are the strings.

B. Training and Testing

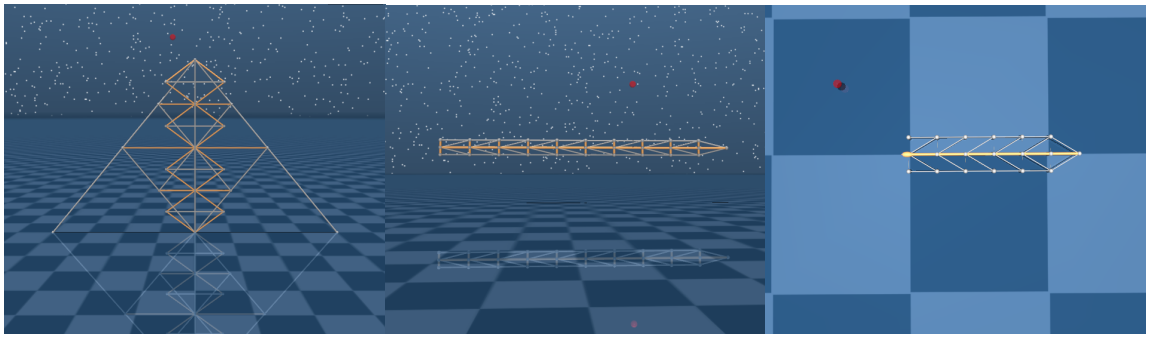
D2C implementation is done in three stages corresponding to those mentioned in the previous section and 'MuJoCo Pro 200 C++ version' is used as the environment for simulating the blackbox model. The model based control (MBC) approach does not require any training time since it is a closed-form solution.

Training (D2C only): The open-loop training plots in Fig. 3 show the cost curves during training. After the cost curves converge, we get the optimal control sequence that could drive the systems to accomplish their tasks. The respective model positions at the end of the horizon are shown in Fig. 2. The training parameters and outcomes are summarized in (Table I).

Testing Criterion: For the closed-loop design, we proceed with the system identification and feedback gain design step of the D2C algorithm mentioned in the previous section to get the closed-loop control policy. For testing, we compare the performance between the open-loop D2C control policy and the closed-loop D2C control policy under different noise levels. We also compare the D2C closed-loop performance to the MBC design for the Reacher example. The MBC design cannot be applied to the arm or the swimmer because of the moment on the elements and solid-fluid interaction. The open-loop control policy is to apply the optimal control sequence solved in the open-loop training step without any feedback. So the perturbation drives the model off the nominal trajectory and increases the episodic cost as the noise level increases. Zero-mean Gaussian i.i.d. noise is added to every control channel at each step. The standard deviation of the noise is proportional to the maximum control signal in the designed optimal control sequence for D2C. As for MBC, we use the maximum control value in the noiseless nominal control sequence. It must be noted that the range of noise levels (at least up until about 60 % of the maximum nominal control signal) that we are considering here is far beyond what is typically encountered in practical scenarios. As for the criterion for performance, we use episodic cost at each noise level. 500 rollouts are simulated at every noise level tested.

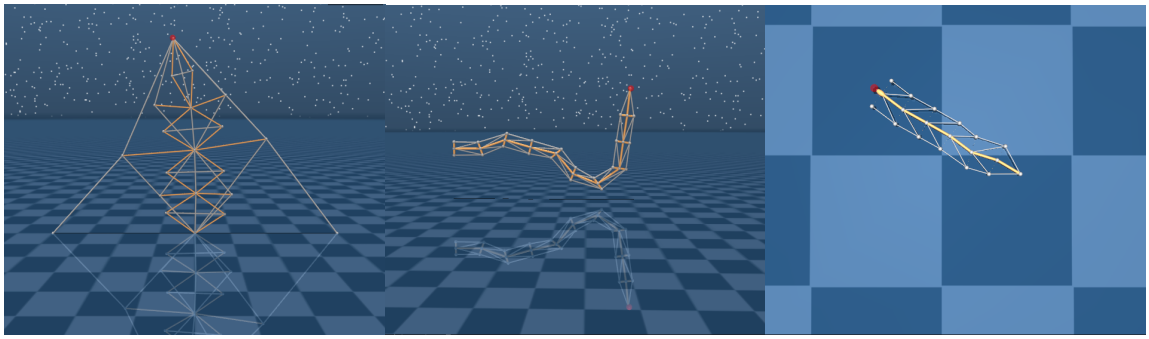
Robustness to Noise: From Fig. 4(a), we can see that both the mean episodic cost and the cost variance of the closed-loop D2C policy is much smaller than that of the open-loop policy for the reacher example throughout the noise level range shown in the plots which prove the success of using D2C on tensegrity models. Although the MBC design can only take up to 30% noise before the simulation fails, it has similar performance with the D2C closed-loop policy when applicable. For the D2C approach, the swimmer example fails after 60% noise, while the reacher and the arm example worked upto 100% noise level. When there is no noise, the analytical solution has the lowest cost of all three. From Fig. 5, it is clear that the control energy used by the MBC solution is way larger than D2C, where the episodic energy is calculated as the L2-norm of the control sequence. This is the result of the optimization-based formulation of D2C. Also, MBC takes up to 30% noise and has higher variance than D2C, which aligns with what is shown in Fig. 4(a). With less control effort, the D2C policy gets a feedback policy close to the accuracy of the analytical solution and endure a wider range of noise. The closed-loop cost for all the three models increases as noise level increases. If we keep increasing the noise level, there exists a threshold that the noise will drive the system too far away from the nominal trajectory that the LQR controller cannot fix it. However, till that point, the closed-loop policy always performs better than the open-loop policy (Fig. 4).

Discussion: The advantage of the MBC approach is that, whenever applicable, it has negligible training time when compared to a data based approach like D2C. However, the D2C approach can achieve the same performance as the model based approach with far less control effort. One of the reasons for this might be that the reference system needed by



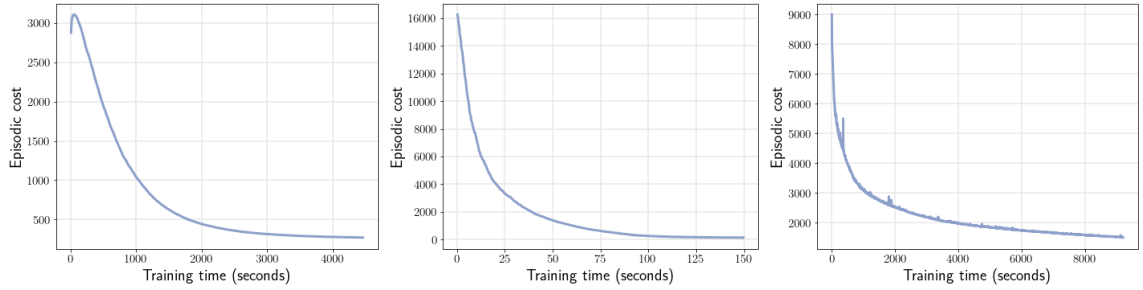
(a) Reacher (b) Arm (c) 6-link Swimmer

Fig. 1: Models simulated in MuJoCo in their initial states



(a) Reacher (b) Arm (c) 6-link Swimmer

Fig. 2: Models simulated in MuJoCo in their terminal states



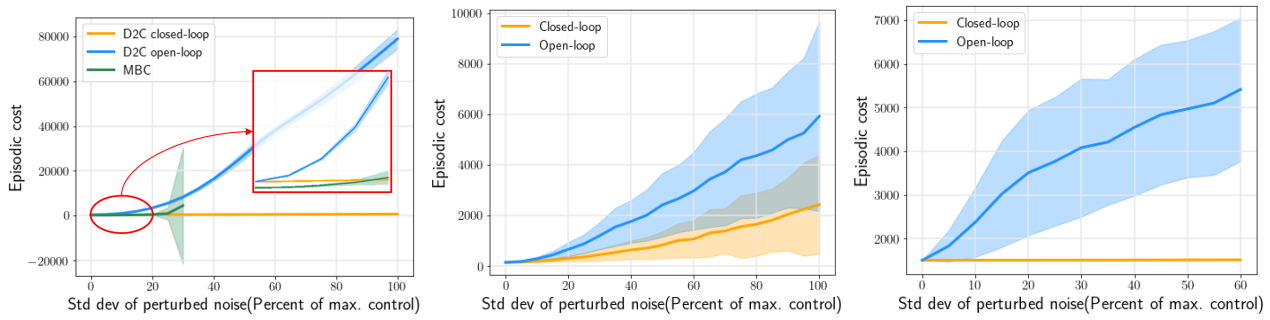
(a) Reacher (b) Arm (c) 6-link Swimmer

Fig. 3: Episodic cost vs time taken during open-loop training

the MBC design has not been designed optimally, and thus, a more careful design might result in control efforts that are on par with D2C. In a sense, the philosophies followed by the D2C and the MBC approach are quite similar in that a reference nominal system to be tracked is designed in both, which is then tracked using the closed-loop control. However, the D2C obtains this via optimizing a suitable finite horizon cost of the nonlinear system whereas the MBC approach provides a somewhat arbitrary prescribed behavior, which, in turn, needs much more control effort in order to be tracked. As mentioned above, the arm and swimmer model are difficult to model due to increased complexity and solid-fluid interaction, but the data based D2C approach can still be applied to these models, indicating a wider application potential for such data based control design approaches.

VI. CONCLUSIONS

In this article, we have provided an overview of the modeling of tensegrity structures that can be used as the mechanism for enabling very high DOF robots, in particular, tensegrity robots. We have also studied model-based and data-based approaches to the control of such robots and have



(a) Reacher (b) Arm (c) 6-link Swimmer

Fig. 4: Performance comparison between D2C open-loop and closed-loop control policy

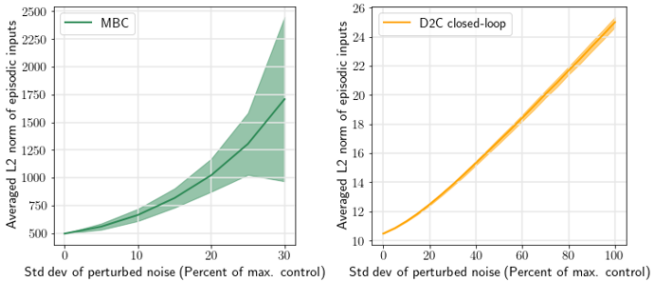


Fig. 5: Control Energy comparison between MBC and D2C

TABLE I: D2C training parameters and outcomes

System	Steps per episode	Time-step (in sec.)	Rollout number	Iteration number	Training time (in sec.)	
					Open-loop	Closed-loop
Reacher	400	0.01	300	1300	4462.1	4.25
Arm	400	0.01	20	600	149.5	4.14
Swimmer	1500	0.006	50	7000	9204.0	5.83

[1] The open-loop training is run on a laptop with I7-7700HQ CPU and 8G RAM. No multi-threading at this point.

outlined their relative merits and demerits. In future work, we shall further develop these control approaches to tensegrity robots, and study the design, construction, and control of such robots. In particular, we plan to apply these techniques to a physical model of tensegrity robotic arm.

REFERENCES

[1] D. Rus, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 5, pp. 467–475, 2015.

[2] C. Majidi, "Soft robotics: a perspective — current trends and prospects for the future," *Soft Robotics*, vol. 1, pp. 5–11, 2014.

[3] R. E. Skelton and M. C. de Oliveira, *Tensegrity Systems*. Springer US, 2009.

[4] R. Goyal and R. E. Skelton, "Tensegrity system dynamics with rigid bars and massive strings," *Multibody System Dynamics*, vol. 46(3), pp. 203–228, 2019.

[5] R. E. Skelton and M. C. de Oliveira, "Optimal tensegrity structures in bending: the discrete michell truss," *Journal of the Franklin Institute*, vol. 347(1), pp. 257–283, 2010.

[6] K. Caluwaerts, J. Despraz, A. Işçen, 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*, vol. 11, no. 98, 2014.

[7] 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)*, May 2017, pp. 5051–5057.

[8] H. Karnan, R. Goyal, M. Majji, R. E. Skelton, and P. Singla, "Visual feedback control of tensegrity robotic systems," *2017 IEEE/RSJ-IROS*, pp. 2048–2053, Sept 2017.

[9] A. G. Tibert and S. Pellegrino, "Deployable tensegrity mast," *In: 44th AIAA/ASME/ASCE, Structures, Structural Dynamics and Materials Conference and Exhibit, Norfolk , VA, USA.*, p. 1978, 2003.

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

[11] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 634–641.

[12] P. R. Kumar and P. Varaiya, *Stochastic systems: Estimation, identification, and adaptive control*. SIAM, 2015, vol. 75.

[13] P. A. Ioannou and J. Sun, *Robust adaptive control*. Courier Corporation, 2012.

[14] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[17] W. Yuhuai, M. Elman, L. Shun, G. Roger, and B. Jimmy, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *arXiv:1708.05144*, 2017.

[18] S. John, L. Sergey, M. Philipp, J. Michael I., and A. Pieter, "Trust region policy optimization," *arXiv:1502.05477*, 2017.

[19] S. John, W. Filip, D. Prafulla, R. Alec, and K. Oleg, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[20] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] R. Wang, K. Parunandi, D. Yu, D. Kalathil, and S. Chakravorty, "Decoupled Data Based Approach for Learning to Control Nonlinear Dynamical Systems," *arXiv e-prints*, p. arXiv:1904.08361, Apr 2019.

[22] D. Yu, M. Rafieisakhaei, and S. Chakravorty, "Stochastic Feedback Control of Systems with Unknown Nonlinear Dynamics," in *56^{th} IEEE Conference on Decision and Control(CDC)*, 2017.

[23] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*. Routledge, New York, 1975.

[24] T. Emanuel, E. Tom, and Y. Tassa, "Mujoco: A physics engine for model-based control," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.