Exploiting the Natural Dynamics of Series Elastic Robots by Actuator-Centered Sequential Linear Programming

Rachel Schlossman\textsuperscript{1}, Gray C. Thomas, and Luis Sentis

Abstract—Series elastic robots are best able to follow trajectories which obey the limitations of their actuators, since they cannot instantly change their joint forces. In fact, their actuators can even allow them to improve over the performance of a robot with an ideal force source actuator by storing and releasing energy. In this paper, we formulate the trajectory optimization problem for series elastic robots in a new way based on sequential linear programming. Our framework is unique in the separation of the actuator dynamics from the rest of the dynamics, and in the use of a tunable pseudo-mass parameter that improves the discretization accuracy of our approach. The actuator dynamics are truly linear, which allows them to be excluded from trust-region mechanics. This causes our algorithm to have similar run times with and without the actuator dynamics. We test the accuracy of our discretization strategy using conservation of energy. We then demonstrate our optimization algorithm by tuning a jump behavior for a single leg robot in simulation, showing that compliance allows a higher jump and takes a similar amount of computation time.

I. INTRODUCTION

Since its inception \cite{1}, a primary drawback of series elastic actuation has been the additional challenge for the control system. Human-centered robots commonly make use of series elastic actuators (SEAs), which offer the benefits of compliance—for safe interaction with humans—increased robustness, and force sensing \cite{2}. The compliant element is able to store and release energy, presenting an opportunity for increased efficiency and agility as compared to rigid actuators \cite{3}. Both feedback controllers and trajectory planners are faced with a more complex challenge when interfacing with these systems, yet modern control systems for human-centered robots (e.g., \cite{4}) rely on a force-control planning abstraction which specifies an unmeetable goal for the low level feedback controller and provides those controllers with planned trajectories that do not respect their dynamic limitations.

Interest in modified series elastic actuators with clutches and variable stiffness compliant elements has driven many groups to derive bang-bang style and cyclic optimal behaviors to illustrate improved mechanical performance \cite{5}. Few groups, however, have investigated more general behaviors which allow for nonlinearities in the system. In \cite{6}, a convex optimization problem is formulated to maximize joint velocity by computing the switching times between rigid and compliant actuator behavior via the use of a clutch, but the actuator dynamics are linear except at switching times. A benefit of our work is the ability to handle the nonlinearities that are introduced at all time steps through a nonlinear transmission, while still leveraging compliance.

Iterative regulator-based optimal control has been successful in handling nonlinearities in these systems and achieving rapid motions in compliant robots, but is restricted in capturing state and input constraints, e.g., transmission speed or spring deflection limits. The iLQR indirect method has been modified to allow input constraints \cite{7}, \cite{8}—but not state constraints directly. In \cite{8}, iLQR is used in combination with variable stiffness actuators to leverage the energy storing capability of the compliant element to throw a ball, but the motor position constraint can only be captured indirectly through the input constraint. Inequality state constraints in \cite{9} are reformulated as canonical input constraints, yet the number of constraints possible with this strategy is at most the number of inputs. In contrast, our work simply lists all linear state and input constraints, which are upheld by the linear program.

Spline-parameterized, nonlinear programming (NLP) and collocation approaches, based on general purpose large-scale

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{(a) Draco leg prototype. Our simulation-only experiments are modeled after this robot, with significantly reduced spring rates, performing the liftoff phase of a jump. (b) Schematics emphasize the nonlinear transmissions between actuator length, $z$, and joint angle, $\theta$, for the ankle and knee. These nonlinear transmissions motivate our choice to represent the robot impedance in actuator-length–actuator-force space rather than the standard joint-angle–joint-torque space.}
\end{figure}
NLP libraries, have been successfully applied to series elastic robots. In [10], optimal walking trajectories are produced via NLP to be consistent with compliant dynamics subject to all relevant constraints with pre-defined contact transition times. [11] adds a collocation method to automatically select contacts, automatically generate multiple steps of walking, and to jump, at the cost of approximating some actuator constraints. This approach leverages powerful and highly general NLP libraries, however these general solvers result in long run-times on the order of an hour, even for problems which have roughly the same number of trajectory parameters as ours.

In this paper, we propose a direct optimization algorithm which efficiently considers the nonlinear effects of the transmission linkages, robot dynamics, input and state constraints, and the energy storing capabilities of the series elastic elements. The algorithm uses sequential linear optimization to minimize a final velocity objective (with a 1-norm input penalty) to demonstrate its ability to produce high performance behaviors while satisfying system constraints. We formulate the problem as input selection for a time-varying discrete time linear system approximation that is updated iteratively. We formulate the system dynamics to connect the actuator space to the joint space. One of the key, novel features of our approach is the use of a fictitious pseudo-mass to improve discretization accuracy for the actuator component at large time steps. Through studying energetic consistency with and without the pseudo-mass, we find this parameter critical in upholding the physics and tractability of our method. By exploiting problem structure via separating the linear and nonlinear components of our model, we typically achieve convergence within 20 iterations.

II. MODELING

A. Actuator Dynamics

Our model considers internal actuator dynamics, which are common for control design, but rare for trajectory design. We follow the advice of [12] and [13], and connect three second-order systems through a differential to develop an unlumped model of the SEA.

The actuator model, as pictured in Figure 2, comprises the spring system; the motor system with input current, \( u \); and the load system. The states considered are spring displacement, \( \delta \); spring velocity, \( \dot{\delta} \); motor displacement, \( y \); and motor velocity, \( \dot{y} \). The variables \( z \) and \( \dot{z} \) correspond to total actuator length and velocity, respectively. The motor subsystem is reflected to prismatic motion through the transmission—hence, all parameters of the subsystems are expressed in linear units. The three systems are connected through a three-way differential, \( D \), which enforces the relationship:

\[
z = \delta + \dot{y}.
\]

The dynamics of the three subsystems are:

\[
M_s \ddot{\delta} + \beta_s \dot{\delta} + k \delta = -f,
\]

\[
M_m \ddot{y} + \beta_m \dot{y} = k_m \dot{u} - f.
\]

\[
(M_L + M_p) \ddot{z} + \beta_L \dot{z} = f - (F - M_p \ddot{z}),
\]

\[
M_m \ddot{y} + \beta_m \dot{y} = k_m \dot{u} - f.
\]

\[
(M_s, M_L, \text{and } M_m \text{ are the masses of the spring, load, and motor systems, respectively; } \beta_s, \beta_L, \text{and } \beta_m \text{ are these systems’ respective damping coefficients; } k \text{ is the spring constant; and } k_m \text{ is the reflected motor constant. The second input, } F, \text{ is the force output from the actuator, which is used to link with the robot dynamics and the nonlinearities in the system. } M_p \text{ is a fictitious pseudo-mass, which will be used to tune the eigenvalues of the linear actuator system before discretization, as discussed in Section III. We define } F’ \text{ as:}
\]

\[
F’ \triangleq F - M_p \ddot{z}.
\]

The variable \( f \) is equal to the back forces from the differential and, equivalently, the Lagrange multiplier which enforces the differential constraint. Substitution for \( f \) reveals that this model is ultimately fourth order:

\[
E_0 \dot{x} = A_0 x + B_{o,u} u + B_{o,f} F’,
\]

where state vector \( x \triangleq [\delta \dot{\delta} \dot{\gamma} \dot{y}]^T \) and

\[
E_0 \triangleq \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & M_s + M_L + M_p & 0 & M_L + M_p \\
0 & 0 & 1 & 0 \\
0 & M_L + M_p & 0 & M_m + M_L + M_p
\end{bmatrix}
\]

\[
A_0 \triangleq \begin{bmatrix}
0 & 1 & 0 & 0 \\
-\beta_s & -\beta_L & 0 & -\beta_L \\
0 & 0 & 1 & 0 \\
0 & -\beta_L & 0 & -\beta_L + \beta_m
\end{bmatrix}
\]

\[
B_{o,u} \triangleq [0 0 k_m]^T, \quad B_{o,f} \triangleq [0 -1 0 -1]^T.
\]

Rearranging (6),

\[
\dot{x} = A_s x + B_{s,u} u + B_{s,f} F’,
\]

where

\[
A_s \triangleq E_0^{-1} A_0, \quad B_{s,u} \triangleq E_0^{-1} B_{o,u}, \quad \text{and} \quad B_{s,f} \triangleq E_0^{-1} B_{o,f}.
\]

\[1\] 1,782 parameters in “less than an hour” [11] versus our 1,176 parameters in 28.5 seconds—iterating an LP 19 times.
From the construction of $A_0$, it is clear that the eigenvalues of $A_1$ will vary with $M_p$. As we proceed, we will discuss the application of this formulation for the general case of $p$ joints. Our state vector will be:

$$x = [x_1^T, x_2^T, \ldots, x_p^T]^T,$$

where each $x_i$ captures the four states described in $q$ for their respective actuator system. Equation $[6]$ is extended (using the Kronecker product $\otimes$) to a $p$-link system with:

$$E_{o,p} = I_p \otimes E_o, \quad A_{o,p} = I_p \otimes A_o, \quad (9)$$

$$B_{o,u,p} = I_p \otimes B_{o,u}, \quad B_{o,F,p} = I_p \otimes B_{o,F}, \quad (10)$$

where $I_p$ is the $p \times p$ identity matrix. Equation $[7]$ can then be reformulated using $[9]$ and $[10]$ to obtain $A_1$, $B_{1,u}$, and $B_{1,F}$ for $p$ joints:

$$\dot{x} = A_1 x + B_{1,u} u + B_{1,F} F'.$$

### B. Robot Dynamics

The force $F$ connects the actuator to the robot dynamics. In general, for a multi-link system, the dynamics are:

$$M(q)\dot{q} + C(q, \dot{q}) + G(q) = \tau = L(q)\dot{F}, \quad (12)$$

where $M$, $C$, and $G$ represent inertia, Coriolis and centrifugal, and gravitational forces, respectively, and $q$ is the generalized joint angle vector.

The angle-dependent moment arm between the actuator and the joint, $L(q)$ abbreviated $L$, serves as the Jacobian between the joint space and the actuator space: $L\dot{q} = \dot{z}$. We solve for $F'$ by projecting this impedance into the actuator-position–actuator-force space:

$$F' = F - M_p \ddot{z} = (L^{-T}M(q)L^{-1} - M_p)\ddot{z} + b(q, \dot{q}), \quad (13)$$

where

$$b(q, \dot{q}) \triangleq L^{-T}(C(q, \dot{q}) + G(q)) - (L^{-T}M(q)L^{-1} - M_p)\dot{L}q. \quad (14)$$

This is an expression for the impedance of the robot at the $\{\dot{z}, F'\}$ port.

### C. Discretization

To prepare for discrete time $u$ optimization, the state space model is discretized into $N$ time steps of length $\Delta T$. By the continuous state space model in $[11]$, acceleration at the actuator output can be computed as:

$$\ddot{z} = S(A_1 x + B_1 \begin{bmatrix} u \\ F' \end{bmatrix}), \quad (15)$$

where $B_1$ is the concatenation of $B_{u}$ and $B_{F}$. This is actuator admittance at the $\{\dot{z}, F'\}$ port. $S$ is formulated to capture the acceleration terms for the $p$-link system:

$$S = I_p \otimes \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}. \quad (16)$$

This formulation gives a value for $F'$ in terms of the states:

$$F' = [I - (L^{-T}M(q)L^{-1} - M_p)SB_{1,F}]^{-1}[L^{-T}G(q) + L^{-T}C(q, \dot{q}) + (L^{-T}M(q)L^{-1} - M_p)(SA_1 x + SB_{1,u} u - L\dot{q})]. \quad (17)$$

We discretize the linear actuator admittance model under the zero-order hold assumption for both $u$ and $F'$. The discrete state space model is then:

$$x_{n+1} = Ax_n + B \begin{bmatrix} u_n \\ F'_n \end{bmatrix}, \quad (18)$$

where

$$A \triangleq e^{A_1 \Delta T}, \quad B \triangleq \int_0^{\Delta T} e^{A_1(\Delta T - \tau)}B_1 d\tau. \quad (19)$$

Combining discrete time admittance and impedance at the $\{\dot{z}, F'\}$ interface, the discretized (time varying) update equation is:

$$x_{n+1} = A_{\text{lin}, n} x_n + B_{\text{lin}, n} u_n + \text{bias}_n, \quad (20)$$

where $A_{\text{lin}}$ and $B_{\text{lin}}$ capture the linear dynamics associated with the actuator states and input current, respectively, and $\text{bias}$ captures the nonlinear robot impedance, including gravity, Coriolis effects, and nonlinear transmissions. The $M_p$ parameter is used to minimize the error introduced by discretizing the actuator admittance in the absence of the reflected inertia of the robot links. This locally-linear model forms the foundation from which our algorithm is developed.

### III. Iterative Linear Programming

For trajectory optimization of a $p$-link system, our approach follows a strategy that culminates in a linear programming subproblem. Our local optimization approach requires a baseline trajectory, $z_{\text{base}}$, to initialize the nonlinear parts of the dynamics. A slow trajectory or a static position both serve as good choices. There is no need for a similar baseline problem. Our local optimization approach requires a baseline trajectory, $z_{\text{base}}$, to initialize the nonlinear parts of the dynamics. A slow trajectory or a static position both serve as good choices. There is no need for a similar baseline trajectory. The $z_{\text{base}}$ trajectory allows us to compute the time varying matrices used in $[17]$ to compute $F'$. We can then compute the linearization components, $A_{\text{lin}}$, $B_{\text{lin}}$, and $\text{bias}$, for each time step (effectively saving our solver from eliminating the $F'$ variable itself).

The linear problem structure can then be exploited. New displacement and velocity trajectories for the spring and motor subsystems are computed via a linear program. The resulting trajectory for $z$ becomes the new $z_{\text{base}}$, and $x^*$ is used to compute the new $F'$, $A_{\text{lin}}$, $B_{\text{lin}}$, and $\text{bias}$ matrices for the next iteration. Trust region constraints will keep the next $z$ trajectory close to this updated $z_{\text{base}}$ trajectory. The algorithm continues to run until the 2-norm of the difference between the current and previous trajectories stops changing.

A key benefit of our approach is that all relevant actuator state and input constraints can be included in the formulation. The constraints are associated with the upper and lower bounds of the allowable spring deflections, $\delta$, joint limits, actuator ballscrew velocity, $\dot{y}$, and input currents, $\pi$. The
parameter $\Delta \bar{z}$ defines the trust region, which can be used to aid convergence of the iteration scheme. We note that the dimension of this trust region is small relative to the full dimension of $x$—again due to separation of the linear and nonlinear dynamics. The final state can be subject to partial end point constraints. Our linear subproblem minimizes a linear cost function, $h(x, u)$, which is a function of, and subject to constraints on, the discretized states and inputs:

$$\begin{align*}
\text{minimize} & \quad h(x, u) \\
\text{subject to} & \quad \text{dynamics:} (20) \\
& \quad \text{a trust region:} \\
& \quad |z_{i,n} - z_{i,n,\text{base}}| \leq \Delta \bar{z} \quad \forall \ i \in \mathcal{P}, \ n \in \mathcal{N} \\
& \quad \text{state constraints:} \\
& \quad |\delta_{i,n}| \leq \delta \quad \forall \ i \in \mathcal{P}, \ n \in \mathcal{N} \\
& \quad z_{\text{min},i} \leq z_{i,n} \leq z_{\text{max},i} \quad \forall \ i \in \mathcal{P}, \ n \in \mathcal{N} \\
& \quad |\gamma_{i,n}| \leq \gamma \quad \forall \ i \in \mathcal{P}, \ n \in \mathcal{N} \\
& \quad |u_{i,n}| \leq \eta \quad \forall \ i \in \mathcal{P}, \ n \in \mathcal{N}/\mathcal{N} \\
& \quad \text{and problem-specific constraints, in our case:} \\
& \quad x_0 = x_{\text{init}}, \ z_N = z_{\text{fin}}. \\
& \quad J_{\text{com}, x, \text{velocity}} \dot{z}_N = 0 \\
& \quad \Phi_{i,n} \geq 0 \quad \forall \ i \in 4, \ n \in \mathcal{N}/\mathcal{N} \\
& \quad |u_{\text{dev},i,n}| \leq u_{\text{abs},i,n} \quad \forall \ i \in \mathcal{P}, \ n \in \mathcal{N}/\mathcal{N} \\
\end{align*}$$

where $|a| \leq b$ is shorthand for two linear inequalities, $-b \leq a \leq b; \forall \ n \in \mathcal{N}$ means $n = 1, \ldots, N; \forall \ i \in \mathcal{P}$ means $i = 1, \ldots, p; \text{and} / \text{means omitting an element from the set. The parameter } \Phi \text{ refers to the foot contact constraints, which will be discussed in Section IV-A. The specific cost function used in our study and problem-specific constraints are described in Section IV-A.}$

A. Algorithm Tuning and Robustness

To achieve convergence, we choose $\Delta \bar{z}$ to limit planning to the region where our linearized dynamics are not too inaccurate. Our novel approach is to select $M_p$ so that the fastest eigenvalue (presumed to be the oscillation one) of $A_{lin}$ and $A_1$ approximate the fastest eigenvalue of the continuous system, ensuring an accurate approximation of the system dynamics. When $M_p = 0$, the spring dynamics settle faster than one time step and cannot be leveraged.

Each iteration of our algorithm relies on the optimal trajectory from the previous iteration, $x_{j-1}^*$, to generate $F_1$, $A_{lin}$, $B_{lin}$, $\text{bias}$, and the subsequent optimal trajectory, $x_j^*$. Hence, without attention to robustness, an infeasible solution will lead to algorithmic failure. We use Algorithm 1 to resolve this issue. Essentially, if the program becomes infeasible, we reduce the step size to maintain problem feasibility in order to handle the more nonlinear regions of the trajectory space.

### Algorithm 1: Trajectory Feasibility Check

1. **if** $\text{Status} == \text{Feasible}$ **then**
2. \[ x_{last\_success\_input} = x_{j-1}^* \]
3. \[ x_{last\_success\_output} = x_j^* \]
4. \[ x_j^* = x_{last\_success\_output} \]
5. **else**
6. \[ x_j^* = 0.5 * (x_{last\_success\_input} + x_j^*) \]

IV. APPTRONIK™ DRACO-INSPIRED SYSTEM

The formulation in the previous section is applied to the two-link Draco robot (Figure 1) in simulation. The Draco humanoid robot leg prototype is driven by viscoelastic actuators at its ankle and knee joints. Because the viscoelastic actuators used in the Draco system are very stiff, approximately $86^6 \text{ N/m}$, these elements offer minimal energy-storing capabilities. For this study, we explore the advantages of implementing softer springs in this system for a high-performance task.

The state space model in (20) is used with $p = 2$. The two actuators have equivalent spring, motor, and load dynamics. The Draco leg, excluding the actuation linkages, is essentially a two-link manipulator, the dynamics of which are available in standard references. The process to develop the dynamic equations of the robot to include the actuator states follows that described in Section III. The variables $F_1'$ and $F_2'$ are obtained from Lagrangian dynamics of $M(q) \in \mathbb{R}^{2 \times 2}$, and $C(q, \dot{q})$, $G(q) \in \mathbb{R}^2$. Due to space limitations, the coefficients of $M(q)$, $C(q, \dot{q})$, and $G(q)$ can be found in [14], and our code is available in the author’s public repository [15].

To use (17), the moment arms, $L_1(q_1)$ and $L_2(q_2)$, of the ankle and knee joints, respectively, must be considered such that:

$$L \triangleq \begin{bmatrix} L_1(q_1) & 0 \\ 0 & L_2(q_2) \end{bmatrix}.$$  

We chose to demonstrate our algorithm for the goal of maximizing velocity at the center of mass (COM) of the robot, to obtain an optimal trajectory for a jumping motion. The parameters used for the simulation were guided by system identification of our lab’s SEA and the parameters of the Draco leg. Select parameters are included in Table I. In the table, the parameters $I_1$, $I_2$, $m_1$, and $m_2$, and $l_1$ and $l_2$ equal the moments of inertia, masses, and lengths of the lower and upper legs, respectively.

A. Ground Contacts

Ground contact wrenches are considered in the Draco model in the styles of [16] [17]. Point contacts with static Coulomb friction, with the coefficient of friction, $\mu = 0.8$, are applied: one at the front of the foot and one at the heel. Friction cones are formulated at each contact point using the basis vectors $b_1 = [\mu \quad 1]^T$ and $b_2 = [-\mu \quad 1]^T$. The positive force intensity parameters $\Phi_1$, $\Phi_2$, $\Phi_3$, and $\Phi_4$ are the basis vector multipliers, with two of these force intensities associated with each end of the foot, as shown in

2In our 2D simulations, this style of linear parameterization is not an approximation of the true friction cone, but it is in 3D space.
A. Velocity Maximization

In this study, the cost function to be minimized expresses the goal to maximize the upward y-velocity of the robot COM at the final time, \( V^+ \triangleq J_{\text{com,y,velocity}} \ddot{z}_N \), where this Jacobian is known a-priori due to our constrained final position, \( z_{\text{fin}} \). We also strive to avoid unnecessary deviations from the equilibrium motor current, and amend the cost function (to be minimized) to include the 1-norm of deviation from the baseline control signal, \( u_{\text{dev}} = u - u_{\text{baseline}} \). However, to keep the cost function linear, we create the variable \( u_{\text{abs}} \) to represent \( |u_{\text{dev}}| \), as shown in (21). We have also added a slight preference towards solutions with small force intensities:

\[
\begin{align*}
  h(x, u) &= -J_{\text{com,y,velocity}} \ddot{z}_N + \alpha \sum_{i \in P} \sum_{n \in N/N} u_{\text{abs},i,n} + \\
  &+ \gamma \sum_{i \in P} \sum_{n \in N/N} \Phi_{i,n},
\end{align*}
\]  

where \( \alpha \) equals \( 1e^{-5} \) and \( \gamma \) equals \( 1e^{-8} \). This cost function is linear, supporting our problem structure.

For our simulation, the initial condition is at equilibrium with the two springs, which drives the formulation of \( z_{\text{base}} \). The initial and final conditions capture that the leg position starts and ends at the same angular configurations, \( q_{1N} \) and \( q_{2N} \). The final constraint is that the x-component of velocity at the COM is equal to zero at the final time.

The sequential linear optimization problem is solved using the Matlab CVX library [18] with the Gurobi solver [19]. A time period of 0.798 s is considered. The algorithm converges in 19 iterations, \( j = 19 \), within a tolerance of 0.001 for \( ||x_j^* - x_{j-1}^*||_2 \). The corresponding behavior is shown in Figure 3. An optimal value of 1.92 m/s upward velocity is achieved. One will notice spring oscillations, demonstrating the use of the two springs to store and release energy. Draco bends down and springs upward, following a jumping trajectory. Figure 6a demonstrates exponential convergence of our iteration scheme.

Because the time varying robot impedance at \( \{\dot{z}, F'\} \) is captured in \( F' \), our model can also be simulated directly for rigid reference. Specifically, (6), (9), and (10) are used without considering the spring subsystem. The cost function in (23) is used for the rigid and compliant cases, and the resulting optimal velocities are compared. Figure 4 shows that the optimal velocity in the compliant configuration, 1.92 m/s, is 16% greater than that of the rigid configuration, 1.65 m/s. For the rigid robot, the ball screw limits are not reached, but its motion is still constrained by acceleration.
and we select an A (b) The eigenvalues of the continuous system, study as a baseline, the results suggest that the error decays exponentially.

Fig. 6. (a) With the 25th iteration trajectory from the compliant jumping study, the results suggest that the error decays exponentially. (b) The eigenvalues of the continuous system. $A_1$, and $A_{1lin}$ vary with $M_p$, and we select an $M_p$ value where the natural frequencies align at 35 Hz.

limits, damping, and ground contact constraints. Figure 5b shows the two configurations in the associated Matlab simulation. Considering the same initial heights of the robots’ COMs, the compliant leg’s COM reaches a height that is 36% higher than that of its rigid counterpart. These results demonstrate the gains that can be achieved from leveraging the dynamics of the springs.

Our optimization program for the rigid system converges in 25 iterations, as compared to 19 iterations in the compliant simulation. Table IV shows the breakdown of average computation time per iteration to calculate $F'$, $A_{lin}$, $B_{lin}$, $bias$, and the wrench components, and the time spent in the Gurobi optimizer. These results demonstrate that the consideration of compliance introduces only slightly increased computational costs in our method.

Our algorithm converges as $\alpha$ is tuned, demonstrating the limits on optimal upward COM velocity, $V^*$, as the penalty increases.

### B. Zero Input Behavior

To validate simulation accuracy, we considered the situation in which no current was sent to the system. We ensured that energy was conserved throughout the simulation. A test was conducted in which the system was released from rest from a nearly-vertical position. Compared to the jumping studies, the system was more heavily influenced by the changing transmission as the links fell downward due to gravity. As shown in Figure 4, the algorithm converged quickly, in 12 iterations, even in this highly nonlinear case, thereby demonstrating its success in handling nonlinearities in the system.

To investigate energy conservation, a time period of 0.60 seconds is considered with $\Delta T = 10^{-4}$ s to allow comparison to the numerically-difficult $M_p = 0$ case. A pseudo-mass value, $M_p = 580$ kg (reflected inertia from the robot space to the actuator space through the nonlinear transmission), results in 0.55% total energy fluctuation during falling, whereas $M_p = 0$ results in 2.73%. The selection of $M_p$ is discussed in Section V-C. With the time step used in the jumping trajectory generation, $\Delta T = 0.0095$ s, energy varies by 1.79% with the reasonable pseudo-mass. As $M_p$ approaches 0, the robot does not even fall, which reflects a loss of important dynamics in the discretization process with this larger time step. At $M_p = 20$ kg and this same time step, energy fluctuates by 46%. These outcomes emphasize the importance of the $M_p$ parameter in designing a reasonably discretized model.

### C. Algorithm Tuning

The spring oscillation eigenvalue of the actuator system is influenced by the reflected link inertia, and can exceed the sampling rate (and therefore suffer from aliasing when discretized) in the absence of a reasonably tuned pseudo-mass parameter. $M_p$ is set to 580 kg for the two actuators based on closeness to the largest eigenvalue in an approximate trajectory (Figure 6b). Since the time step $\Delta T$, and associated sampling frequency used to discretize the system in Section I-A must be significantly greater than the largest eigenvector of the continuous system to avoid aliasing, the pseudo-mass modification is essential to allowing large time-steps, small linear program sizes, and fast run-times.

We also experimented with providing our algorithm with a better-informed nominal trajectory. Specifically, we used the optimal trajectory of the rigid system as the nominal trajec-

| Configuration | Linearization | Optimization | Total Time |
|---------------|---------------|--------------|------------|
| Compliant     | 0.077         | 1.32         | 28.5       |
| Rigid         | 0.072         | 1.14         | 32.1       |
tory for the compliant leg. With the same tolerance (0.001), the algorithm converges in 17 iterations rather than 19. This result supports only a small efficiency gain from using rigid system trajectory optimization as a warm start heuristic for actuator-centric trajectory optimization. We expect that in practice, actuator-less kinematic trajectory planning will be a critical first step even if actuator-less trajectory optimization is not a useful warm-start.

Figure 7 shows the results of modifying the input penalization parameter, $\alpha$, as defined in (23). The results show the expected decrease in optimal velocity after reaching a threshold value, as well as success in algorithm convergence for various tuned cost functions.

VI. DISCUSSION

Our proposed method for trajectory optimization offers several advantages. First, directly capturing all relevant state and input constraints is an essential feature for a dynamically consistent trajectory. Our new robot–actuator interface, modified by pseudo-mass $M_p$, allows us to exploit the structural difference between a linear actuator admittance and a nonlinear robot impedance—which is novel and efficient. Through formulating a linear subproblem, we take advantage of the computational efficiency of solving these problems, while handling nonlinearities. In our simulations, the algorithm typically converges in about 30 seconds. Finally, we have demonstrated the gains in executing a high-performance task by leveraging compliance in the linear optimization subproblem. The performance of our approach depends strongly on the subproblem solver. While CVX offers several solvers, we found that using a solver specifically suited for linear programming, Gurobi, was needed to extend our approach to larger time scales.

In the future, we are interested in how our actuator model could add on to any standard robot impedance model, including the very general constrained floating base model. As actuators cannot function as perfect torque sources, planners that have the knowledge of the actuators’ more-detailed abilities will allow them to produce achievable trajectories which can leverage the natural dynamics endowed by their low level components.

ACKNOWLEDGMENT

The authors would like to thank Junhyeok Ahn, Orion Campbell, Donghyun Kim, Jaemin Lee, Nicholas Paine, and Tom Gannon for their support.

REFERENCES

[1] G. A. Pratt and M. M. Williamson, “Series elastic actuators,” in Intelligent Robots and Systems 95. Human Robot Interaction and Cooperative Robots. Proceedings. 1995 IEEE/RSJ International Conference on, vol. 1. IEEE, 1995, pp. 399–406.
[2] N. Paine, S. Oh, and L. Sentis, “Design and control considerations for high-performance series elastic actuators,” IEEE/ASME Transactions on Mechatronics, vol. 19, no. 3, pp. 1080–1091, 2014.
[3] K. Sreenath, H.-W. Park, I. Poullakakis, and J. W. Grizzle, “A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL,” The International Journal of Robotics Research, vol. 30, no. 9, pp. 1170–1193, 2011.
[4] J. Pratt, T. Koolen, T. De Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, “Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid,” The International Journal of Robotics Research, vol. 31, no. 10, pp. 1117–1133, 2012.
[5] B. Vanderborght, A. Albu-Schaffer, A. Bicchi, E. Burdet, D. G. Caldwell, R. Carlston, M. Catalano, O. Eiberger, W. Friedl, G. Ganesh et al., “Variable impedance actuators: A review,” Robotics and autonomous systems, vol. 61, no. 12, pp. 1601–1614, 2013.
[6] L. Chen, M. Garabini, M. Lafranchi, N. Kashihi, N. G. Tsgarakis, A. Bicchi, and D. G. Caldwell, “Optimal control for maximizing velocity of the compact compliant actuator,” in Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013, pp. 516–522.
[7] A. Radulescu, M. Howard, D. J. Braun, and S. Vijayakumar, “Exploiting variable physical damping in rapid movement tasks,” in Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on. IEEE, 2012, pp. 141–148.
[8] D. Braun, M. Howard, and S. Vijayakumar, “Optimal variable stiffness control: formulation and application to explosive movement tasks,” Autonomous Robots, vol. 33, no. 3, pp. 237–253, 2012.
[9] D. J. Braun, F. Petit, F. Huber, S. Haddadin, P. Van Der Smagt, A. Albu-Schäffer, and S. Vijayakumar, “Robots driven by compliant actuators: Optimal control under actuation constraints,” IEEE Transactions on Robotics, vol. 29, no. 5, pp. 1085–1101, 2013.
[10] A. Werner, B. Henze, F. C. Loeffl, S. Leyendecker, and C. Ott, “Optimal and robust walking using intrinsic properties of a series-elastic robot,” in IEEE-RAS International Conference on Humanoid Robots, 2017.
[11] A. Werner, W. Turlej, and C. Ott, “Generation of locomotion trajectories for series elastic and viscoelastic bipedal robots,” in IEEE International Conference on Intelligent Robots and Systems, 2017.
[12] V. L. Orekhov, C. S. Knabe, M. A. Hopkins, and D. W. Hong, “An unlumped model for linear series elastic actuators with ball screw drives,” in Intelligent Robots and Systems (IROS). 2015 IEEE/RSJ International Conference on. IEEE, 2015, pp. 2224–2230.
[13] S. Schütz, A. Nejadfard, C. Kötting, and K. Berns, “An intuitive and comprehensive two-load model for series elastic actuators,” in Advanced Motion Control (AMC), 2016 IEEE 14th International Workshop on. IEEE, 2016, pp. 573–580.
[14] H. Asada and J. Leonard, 2.12 Introduction to Robotics. Fall Massachusetts Institute of Technology: MIT OpenCourseWare, License: Creative Commons BY-NC-SA, 2005. [Online]. Available: https://ocw.mit.edu
[15] R. Schlossman. https://github.com/rschloss123 2018.
[16] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, “Design of a momentum-based control framework and application to the humanoid robot atlas,” International Journal of Humanoid Robotics, vol. 13, no. 01, p. 1650007, 2016.
[17] G. C. Thomas and L. Sentis, “Towards computationally efficient planning of dynamic multi-contact locomotion,” in Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016, pp. 3879–3886.
[18] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 2.1,” Mar. 2014.
[19] I. Gurobi Optimization. “Gurobi optimizer reference manual,” 2016. [Online]. Available: http://www.gurobi.com.

[20] I. Gurobi Optimization, “Gurobi optimizer reference manual,” 2016. [Online]. Available: https://www.gurobi.com