Blended linear models for reduced compliant mechanical systems

Sheldon Andrews, *Member, IEEE,* Marek Teichmann, Paul G. Kry

Abstract—We present a method for the simulation of compliant, articulated structures using a plausible approximate model that focuses on modeling endpoint interaction. We approximate the structure's behavior about a reference configuration, resulting in a first order reduced compliant system, or FORK^{-1}S. Several levels of approximation are available depending on which parts and surfaces we would like to have interactive contact forces, allowing various levels of detail to be selected. Our approach is fast and computation of the full structure's state may be parallelized. Furthermore, we present a method for reducing error by combining multiple FORK^{-1}S models at different linearization points, through twist blending and matrix interpolation. Our approach is suitable for stiff, articulate grippers, such as those used in robotic simulation, or physics-based characters under static proportional derivative control. We demonstrate that simulations with our method can deal with kinematic chains and loops with non-uniform stiffness across joints, and that it produces plausible effects due to stiffness, damping, and inertia.

Index Terms—character animation, physics simulation, constraints

1 Introduction

Real-time physics simulation has emerged as a fundamental component of interactive immersive virtual environments. There are important applications in operator training for robots and heavy equipment, robotic design, and simulation of virtual humans in video games.

In this paper, we describe a technique that improves interactive simulations involving complex multi-body mechanisms with contact. For instance, the simulation of human or robotic hands during grasping involves complicated chains of compliant joints and distributed contacts. Collaborative grasping and manipulation with multiple people, multi-legged robots, and vehicle suspension systems can produce similarly difficult computational scenarios. Simulating these kinds of systems is tricky because they result in large over-constrained systems of equations that may require considerable computational effort to solve. Furthermore, special attention must be paid to the parameters of both the system and simulation to ensure stability. Our technique simplifies these kinds of systems, allowing for complex mechanisms to be simulated while meeting real-time requirements.

Our approach is based on two main assumptions. The first assumption is that there are a limited number of surfaces at which articulated systems experience contact. Thus, we focus on the effective mechanical properties of a small collection of bodies in the system. This is a reasonable assumption for many scenarios, such as the wheel ground contact for a vehicle, the fingertips of a hand during grasping, or simulated tool-use by virtual humans and robots. The second assumption is that the multi-body system has a reference pose, which is held due to linear springs at the joints. This is certainly true for systems that have passive linear elastic joints, but also reasonable for virtual humans following the equilibrium point hypothesis of motor control, and in simulated robots using proportional derivative (PD) control.

In the case of a single interaction surface, our approach simplifies an entire system to a single 6D mass-spring system. When there are multiple bodies with surfaces experiencing contact, we use a collection of compliantly coupled 6D bodies. We present an incremental algorithm that walks the body connectivity graph to compute the dynamics model. The result is a system that is much simpler than the original, and is also stable and fast to compute. Note that we do not assume that the structure has the topological structure of a tree, as is necessary for fast computation in many alternative multi-body algorithms.

We call the resulting model a *first order reduced compliant system*, abbreviated as FORK^{-1}S. The method description and model construction originally described by Andrews et al. [1] are presented in this article with additional numerical details and with numerous edits focused on improving clarity (for example, the use of compliance matrices instead of inverse stiffness). In this article we also present an important extension to the original FORKS model. We show how to combine multiple FORK^{-1}S models at different linearization points to reduce simulation errors over a wider range of the state space.

For a single FORK^{-1}S model at simulation time, external forces produce a dynamic transient behavior for the bodies that we include in the model. The positions of all the remaining bodies are visualized by computing a compatible state. Rather than using inverse kinematics, we compute linear maps that provide twists for each body as a function of the reduced system state. With these maps, we then use the exponential map to compute the body positions. Thus, non-interacting body positions can be computed independently and in a parallel fashion. Although the twists only model the linear response, we observe that the exponential map gives good behavior with little separation at joints for an

• *Sheldon Andrews and Paul G. Kry are with the School of Computer Science, McGill University, Montreal, QC, Canada. Marek Teichmann is with CMLabs, Montreal, QC, Canada.*

Manuscript received March, 2015.

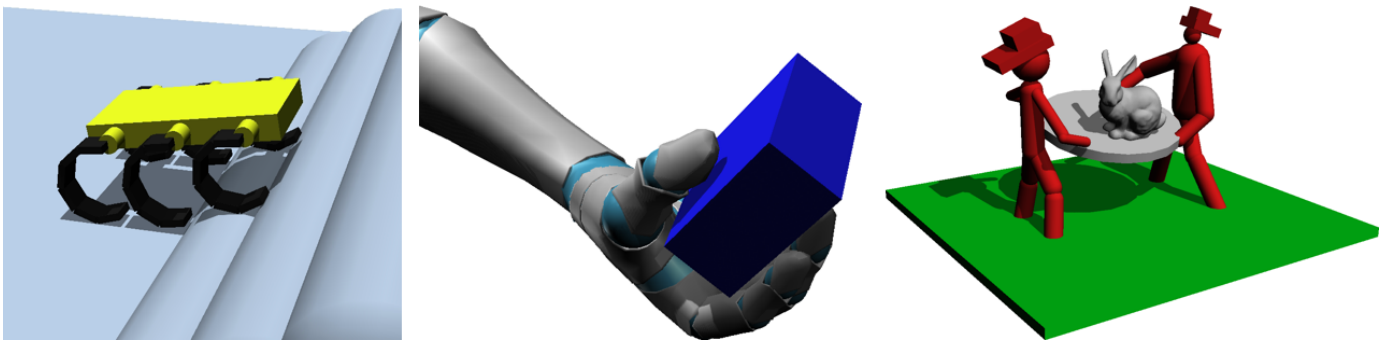

Fig. 1. Example simulations that we use to demonstrate our technique include a multi-legged robot on uneven terrain, two firefighters catching a bunny, and humanoid robot grasping. The reduced model for the firefighters involves only one body, the trampoline, while the reduced models for the legged robot and hand involve multiple coupled end effectors.

adequate range of interaction forces, and we discuss the size of the errors produced. However, there will always be states with visible error when using only a single model, thus we describe in this article how to use multiple models to reduce the kinematic reconstruction errors as desired. In particular, the collection of linearized models can be built by an iterative and interactive online process, by computing and adding a new FORK^{-1}S model whenever the current collection does not provide a suitable simulation at a given point in the state space.

Because the position updates can all happen in parallel, our method is well suited to parallel implementation on modern CPUs and GPUs. With new hardware primarily gaining additional computation power through increasing core counts, we believe it is important for future algorithms to exploit parallelism, and we identify this separable computation of the internal state as one of the important contributions of our work. Additionally, note that the runtime computation necessary to prepare the results of different models for blending can also be done in parallel.

Another important aspect in our work is that we have control over the fidelity of the physics simulation, and can dial it up or down as necessary. For instance, we can simplify a grasping system to be the finger tips of a hand in frictional contact with a grasped object. Alternatively, assuming no sliding or rolling at contacts, we can reduce the system to model the grasped body alone, which may be of interest in the simulation of a peg-in-hole insertion task. Likewise, if we know that additional contacts will occur at other bodies in the multi-body system, then we can include those interaction surfaces in our model. Furthermore, in contrast to our previous work [1], we can reduce simulation errors over a wider range of the state space by combining multiple FORK^{-1}S models at different linearization points, as we detail in this article. This is not simply to reduce kinematic errors, but also to better model variations in the effective stiffness of the model throughout the state space.

We demonstrate various scenarios that show the utility of our approach and the models that we can produce, such as simulation of human grasping and multi-legged robots, as shown in Figure 1. We provide computation time comparisons, discuss approximation errors and limitations of the model. Finally, we identify a few interesting opportunities for future work.

2 Related work

Modeling and animating physical systems at different levels of fidelity is a common objective in many aspects of physics-based animation, such as simulating deformation, friction, contact, and collision. For deformation, there has been a vast amount of work in computer graphics exploiting modal vibration models for reduction. A good survey can be found in a state of the art report by Nealen et al. [2], while other alternative elastic simulation reduction techniques continue to be an active area of research [3], [4], [5], [6]. Frictional contact computations can also be simplified in a variety of ways, such as exact Coulomb friction cones, discretized friction pyramids, box constraints, or penalty based methods [7], [8], [9]. With respect to the contact equations, the contact patches can be discretized at arbitrary resolutions [10]. Finally, collision detection and response can be modified to produce various plausible animations with different fidelity levels [11].

One approach for the simulation of multi-body mechanical systems is to use a constrained full-coordinate formulation. Such systems can be solved quickly with sparse methods, and linear time solutions are possible when the structure has the connectivity of a tree [12]. Constrained multi-body simulations are popular for their simplicity, and are available in a number of different software libraries including Vortex, PhysX, Havok, DART, and the Open Dynamics Engine. Numerical drift must be addressed in this case using stabilization techniques [13], and loops result in redundant constraints that require additional attention in the solution of such systems [14], [15]. Recently, Tomcin et al. [16] describe an efficient solution to systems with redundant constraints by using Tikhonov regularization with carefully selected parameters.

The alternative to full-coordinates is to formulate the system in minimal coordinates, i.e., the joint angles [17]. Straightforward linear time solvers have been used for decades, and divide and conquer approaches permit parallel algorithms with log time complexity [18], [19]. Mechanical structures with loops likewise require special treatment and modified solvers. Various libraries based on minimal coordinates exist, such as SD/FAST which is commonly used in mechanical engineering applications, and DART which is designed specifically for character animation and simulated robots.

Our approach resembles neither a minimal coordinate or redundant coordinate constrained multi-body system. Instead, our first order reduced compliant systems more closely resemble coupled elastic mass-spring systems. We note that other approaches have been proposed for reducing multi-body dynamics. For example, adaptive dynamics is possible in reduced coordinates through rigidification of selected joints [20]. In contrast, with a constrained redundant coordinate formulation, modal reduction of rigid articulated structures is possible and has been used to animate character locomotion [21], [22].

We observe that inverse kinematics techniques would also be a solution for determining the positions of internal parts of the mechanism. There are a variety of fast methods for solving over-constrained inverse kinematics problems using singularity-robust inverse computations [23] or damped least squares [24]. Instead, we opt for the computation of each body individually with a twist because each can be updated in parallel, providing a constant time solution. While typically unnoticeable for small motions, we evaluate how joint constraint violations grow and discuss options to minimize or avoid visible errors during larger interaction forces by interpolating multiple linear models.

End effector equations of motion are important in robot control and analysis, and projections of system dynamics are a central part of the operational space formulation of Khatib [25]. Our incremental projection of the dynamics produces a similar model. In contrast, effective end-point dynamics can also be estimated from data. For instance, model fitting has been applied to human fingertips and hands [26], [27]. In our case, a dynamics projection approach is preferable because the fitting process can become difficult when data is complex. Fitting a simple 6D linear mass-spring is undesirable when the force to displacement relationship in the full model exhibits non-linearities and bifurcation behavior. Furthermore, sampling the system behavior can be expensive and does not fit the desired work flow of interactive simulators. However, it is not unreasonable to impose such a simple model, and to identify its behavior based on a projection of linear compliant or PD controlled behavior of joints. Ultimately, our simplification produces an inexpensive first order model with a plausible response corresponding to a slightly modified set of non-linear joint controllers.

We use concepts of 6D rigid motion in this paper. The book by Murray et al. [28] includes a good overview of the mathematics of rigid motion, twists, wrenches, and adjoint transformations between different coordinate systems. We also provide a brief introduction to rigid body kinematics and related definitions in the Appendix.

Finally, note that this work extends that of Andrews et al. [1]. We include in this article the formulation of the FORK^{-1}S model, but we make a number of improvements for clarity. For instance, compliance matrices are not denoted as an inverse of a stiffness in this article so as to avoid the confusion when the stiffness is not invertible. We add some additional explanations and adjust notation in Sections 3, 4, and 5. Our extensions to multiple linearized models are described in Section 7, and the improvements that this brings are evaluated and discussed in Section 8.

3 Dynamics Projection and Notation

Our method targets mechanisms made up of articulated chains of compliant joints where external interaction occurs only with a small collection of bodies at the interface. We call these bodies the *end effectors*, following the terminology used in robotics literature. For simplicity, we initially present our approach for the case where the base, or root, of the mechanical system is fixed in the world (the case of a free-body base link is discussed in Section 6). There are numerous simulation and animation applications where this type of configuration occurs, such as the grasping and manipulation examples described in Section 1.

In this section, we first explore the simple case of dynamics projection for a single body with one rotational joint, and provide a preliminary discussion of how we can incrementally build a projection for a complex mechanical system.

3.1 Projection for a single link

Consider the behavior of a rigid body link x rotating about a hinge joint, as illustrated by Figure 2. For simplicity, we assume the hinge constraint is fixed in the global coordinate frame. The hinge constrains the motion of the body to a single degree of freedom and the admissible twists are rotations about the joint axis. Therefore, the twist of the rigid body $\xi \in \mathbb{R}^6$ is a function of the joint angle $\theta \in \mathbb{R}$, or $\xi = f(\theta)$. The Taylor expansion of $f(\theta)$ gives the approximation

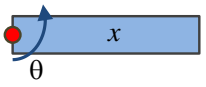

Fig. 2. A single rigid body link rotating about a compliant joint.

$$\xi \approx f(0) + \frac{\delta f}{\delta \theta}\Delta\theta, \tag{1}$$

where $J = \frac{\delta f}{\delta \theta}$ is the Jacobian of the kinematic configuration of the body. Without loss of generality, let $f(0) = 0$, giving the first-order kinematic relationship $\xi \approx J\Delta\theta$.

Assuming a stiff joint, any displacement of the hinge from its rest or initial configuration will generate a torque about the joint axis. The torque τ and joint displacement $\Delta\theta$ of the hinge are related by

$$C_\theta \tau = \Delta\theta, \tag{2}$$

where the scalar $C_\theta = K_\theta^{-1}$ is the compliance, or inverse stiffness, of the joint. This is why we abbreviate the name *first order reduced compliant systems* as FORK^{-1}S.

Body wrenches $w \in \mathbb{R}^6$ corresponding to joint torques are computed by the transpose of the Jacobian,

$$\tau = J^T w. \tag{3}$$

Combining Equations 2 and 3 and multiplying on the left throughout by J gives

$$JC_\theta J^T w = \xi. \tag{4}$$

Here, $C_x = JC_\theta J^T$ is the effective compliance of the body x in spatial coordinates. The twist of the body ξ resulting from an applied external wrench w_{ext} is computed as

$$C_x w_{ext} = \xi \tag{5}$$

and the homogeneous transformation of the body's displacement is computed by the exponential map. Specifically, we denote H_ξ as the 4-by-4 homogeneous transformation matrix of a body displaced by twist ξ from the rest configuration and compute it as

$$H_\xi = H \exp(\xi), \tag{6}$$

where H is the transformation matrix of the body in the the rest configuration (additional comments with respect to this matrix exponential appear in the appendix).

It is convenient to use compliance to model the behavior of the body in full coordinates because this compliance will be zero for motions not permitted by the joint. As such, C_x can be rank deficient, and a robust method for computing the matrix inverse is necessary to compute the stiffness K_x. Our work uses a *truncated* singular value decomposition (SVD) [29] to compute the inverse when needed.

We can perform a very similar projection of the rotational mass matrix M_θ by working with the inverse mass $A = M^{-1}$, which we define as *accelerability*. From the equation of motion $A_\theta \tau = \ddot{\theta}$, and assuming the body acceleration $\dot{\phi} \in \mathbb{R}^6$ is approximately $J\ddot{\theta}$, we find $JA_\theta J^T w = \dot{\phi}$, thus,

$$A_x = JA_\theta J^T. \tag{7}$$

We follow the same projection for the damping matrix to produce the second order system

$$M_x\dot{\phi} + D_x\phi + K_x\xi = w. \tag{8}$$

Note that the static solution of this system exactly matches that of the original. Also notice that this example is more instructional than useful given that Equation 8 has six dimensions while it was constructed from a 1D joint. That is, Equation 8 is rank one in this case and requires that we produce a solution within the space of admissible motions. In contrast, many of the larger more complex examples we show in this paper result in full rank systems.

Despite the simplicity of this example, these projections are useful and a central part of the approach we describe in Section 4. Specifically, the end effector will be part of a complex system of joints and rigid bodies. The effective compliance at the end effector x is due not only to the compliant behavior of the directly attached joint, but also depends on the compliance of its parent (and the rest of the system). As such, we will describe an incremental approach for computing the effective stiffness, damping, and mass, at each link in an articulated system.

3.2 Truncated SVD tolerance

Matrix inversion by truncated SVD is used extensively in our work and this requires tuning a tolerance parameter ϵ. A popular formula [30] for choosing the tolerance value of a matrix $A \in \mathbb{R}^{m\times n}$ is

$$\epsilon = \max(m, n)\, \|A\|\, \varepsilon, \tag{9}$$

where ε is the machine epsilon value. Application of Equation 9 typically results in tolerance values that are suitable for inversion of the effective matrices. However, for some of the more complicated mechanisms in this paper we found it necessary to adjust the tolerance value. This is done by inspection of the mechanism and modifying the tolerance parameter until the rank of the effective matrices matches the expected value. Note that the rank of the effective matrices corresponds to the number of admissible twist directions, and this corresponds to the actual degrees of freedom of the system.

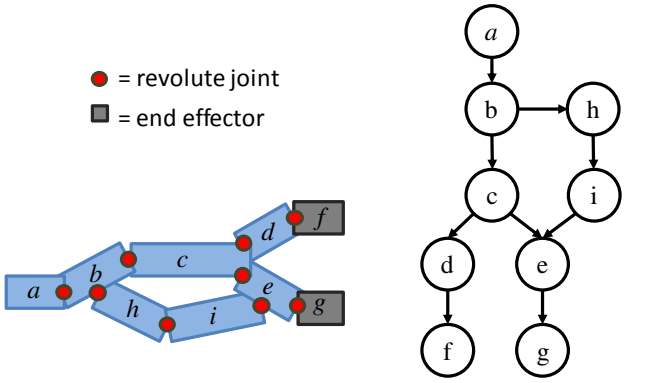

Fig. 3. An image visualizing the connectivity of bodies used in constructing the effective coupled stiffness, damping, and mass of end effectors.

3.3 Notation

Throughout this article we use different coordinate frames. Preceding superscripts are used to denote the coordinate frame in which a vector is expressed, for instance, the velocity $\phi \in \mathbb{R}^6$ of body i in frame j is denoted by ${}^j\phi_i$. Likewise, a wrench acting on body i in frame j is denoted by ${}^jw_i$. The adjoint matrix ${}^k_j\text{Ad}$ maps the twist of a body in frame j to frame k, while its inverse transpose is used to change the coordinates of a wrench from frame j to frame k.

The joint structure of the mechanism has a dual representation as a directed acyclic graph (DAG), as seen in Figure 3. The graph's nodes are links (i.e., rigid bodies), and the edges correspond to joint constraints, with the direction denoting the parent-child relationship. Specifically, the terms *parent link* and *child link* refer to the rigid bodies associated with the outgoing and incoming vertex of the edge, respectively. Using calligraphic font for sets, we write $\mathcal{P}_i$ and $\mathcal{C}_i$ to denote the set of parents and children of link i. Also, $\mathcal{A}_j$ is the set of ancestors of link j where branching occurs (e.g., from Figure 3, $\mathcal{A}_f$ contains c and b). Let $\mathcal{E}$ denote the set of end effectors, and $\mathcal{L}$ denote all the internal links of a mechanism. Note that a body can only belong to one of $\mathcal{E}$ or $\mathcal{L}$.

In Section 7, we describe how to blend multiple models. We use a number in parentheses to denote the model index, for instance, $\xi^{(i)}$ denotes a twist from the ith FORK^{-1}S model. Interpolated quantities are denoted with an over-bar, for example, $\bar{\xi}$ denotes an interpolated twist.

4 Incremental FORK^{-1}S Construction

We incrementally build our approximated model by starting at the base link and working towards the end effectors. The process is simplified by examining three fundamental cases: *chaining*, *splitting*, and *merging*. In Figure 3 we can observe the three cases. Body b is a chained extension from body a. There exists a split at body b because both c and h are children. Finally, the only merge exists at body e, which has the two parents c and i. It is interesting to note that the

Algorithm 1 Recursive algorithm for computation of C. Initiate recursion with RECURSECOMPLIANCE(*base*).

```
function RECURSECOMPLIANCE(i)
 if |P_i| == 1 then
  C_{i,i} ← CHAIN(i)                    // apply Equation 10
 else if |P_i| > 1 then
  C_{i,i} ←MERGE(i)                     // apply Equation 14
 end if
 for j ∈ C_i do
  RECURSECOMPLIANCE(j)
 end for
 if |C_i| > 1 then
  SPLIT(i)                              // apply Equation 12
 end if
end function
```

parent child relationship between bodies c and e can be set in either direction without affecting the final projection.

4.1 Chaining

The effective compliance of body b in the chaining case is the sum of the compliance of the parent link a and the compliance of the joint between the two bodies, C_θ. Assuming the effective compliance of the parent link has already been constructed, it is straightforward to compute the effective compliance of the child link. The twist-wrench relationship at the link is

$$C_{b,b} = JC_\theta J^T + {}^b_a\mathsf{Ad}\, C_{a,a}\, {}^b_a\mathsf{Ad}^T, \tag{10}$$

where $C_{a,a}$ is the effective compliance of the parent link, J and C_θ are respectively the kinematic Jacobian and compliance of the common joint. Multiplying the compliance $C_{b,b}$ by a wrench ${}^b w_b$ produces a twist, where the total twist is the sum of two parts: the twist due to the parent's motion and the twist due to the common joint motion.

Note that we use two identical subscripts to denote the compliance $C_{b,b}$ because we want the motion of body b due to wrenches on body b. In the sections that follow, we will also need to capture the motion of one body due to a wrench on another body in the system. Also notice that the compliance matrix could be written ${}^b_b C_{b,b}$ to denote that it provides twists expressed in the coordinate frame of body b and must be given wrenches expressed in the same coordinate frame. We will drop these preceding scripts when the coordinate frames are clear due to context.

4.2 Splitting

In the case where two or more links share a common parent, their motion is coupled through their common parent. For a link with m children, the linear system determining the twist-wrench relationship of the child links is $\Phi = \mathsf{C}\mathsf{w}$ where

$$\mathsf{C} = \begin{bmatrix} C_{1,1} & \dots & C_{1,m} \\ \vdots & \ddots & \vdots \\ C_{m,1} & \dots & C_{m,m} \end{bmatrix} \tag{11}$$

and $\Phi = ({}^1\phi_1^T \cdots {}^m\phi_m^T)^T$, $\mathsf{w} = ({}^1w_1^T \cdots {}^m w_m^T)^T$. In block matrix C the diagonal block $C_{i,i}$ is the compliance of child link i computed as described for the chaining case, while the off-diagonal blocks provide the coupling. That is, $C_{i,j}$ determines the twist at link i due to a wrench applied to sibling link j. To create these off diagonal blocks, an adjoint transform is used to first map wrenches in link j to the common parent k. The resulting twist is determined by the compliance of the parent, which is then mapped from frame k into the local coordinate frame of link i:

$$C_{i,j} = {}^i_k\mathsf{Ad}\, C_{k,k}\, {}^j_k\mathsf{Ad}^T. \tag{12}$$

Chaining additional links after a split is very similar. The diagonal blocks are updated as per Equation 10, while the off diagonal blocks are updated using Equation 12, where k is the *lowest common ancestor* of the two links i and j.

4.3 Merging

Unlike the cases of serial chain and splitting which work with compliances, merging uses the effective stiffness of the parent links. The effective stiffness is a parallel combination of the effective stiffness of the parent links, and includes the coupled stiffness due to a common ancestor.

The compliant behavior of link k results from multiple coupled chains attached to a single rigid body. For a link with m parents, we consider that the motion of k is the result of multiple superimposed versions of the link, with virtual link labels $1, \dots, m$, and coupled compliance matrix C computed with Equation 11. Let us now write the linear system describing the wrench-twist relationship using the stiffness $\mathsf{K} = \mathsf{C}^{-1}$, as

$$\mathsf{w} = \mathsf{K}\Phi \tag{13}$$

where $\Phi = ({}^k\phi_1^T \cdots {}^k\phi_m^T)^T$ and $\mathsf{w} = ({}^k w_1^T \cdots {}^k w_m^T)^T$. Note that we use coordinate frame k for all blocks, and observe that ${}^k\phi_1 = {}^k\phi_2 = \cdots = {}^k\phi_m$ because the twist motion of all the virtual links must be identical. Also, the accumulation of wrenches equals the total wrench at link k, that is, ${}^k w_k = \sum_{i=1}^m {}^k w_i$. Therefore, the effective stiffness at link k is

$$K_{k,k} = \mathrm{I}\,\mathsf{K}\,\mathrm{I}^T \tag{14}$$

where $\mathrm{I} = (I \cdots I)$, and I is the 6×6 identity matrix. That is, the stiffness is the sum of all the blocks of the inverted coupled compliance matrix. The effective compliance at link k is computed by inverting $K_{k,k}$ using the truncated SVD method.

Instead of creating the coupled compliance incrementally, starting from the base and moving out to the end effectors, we use the recursive approach outlined in Algorithm 1, which combines the chaining, merging, and splitting techniques described in this section. This process is initiated by a single call RECURSECOMPLIANCE(a), where parameter a is the base link of the mechanism.

5 WRENCH AND TWIST MAPS

With the method described in the previous section, we can construct the coupled compliance, damping, and mass matrix of the end effectors. This allows us to simulate a reduced system consisting only of the end effectors. However, we still need to visualize the positions of all links in the structure. For this, we use the static pose twist of internal links as computed from a set of end effectors wrenches, and we call this the *twist map*. At a given time step of the reduced simulation, we compute wrenches that explain the current

state, i.e., current twists, thus producing a compatible pose for the internal links.

In order to compute the twist map, we first describe the construction of a *wrench map* that distributes wrenches applied at the end effectors to the internal links. These maps are built incrementally. Our algorithm starts at each end effector and traverses the DAG of the mechanism in reverse order. First, a local wrench map is found that distributes wrenches from child links to their parents. Then, a global wrench map is computed by a compound matrix transform along the kinematic chain.

5.1 Local wrench map

Given the wrench at link k, the local wrench map may be used to compute the wrenches distributed amongst its parent links. Naively, we could simply divide the wrench by the number of parent links and compute the wrench to transfer to the parent links using the appropriate adjoint transforms. However, this division of force will not be correct because the wrenches transmitted down different chains will depend on the effective compliance of each chain. Thus we construct a linear system to ensure that wrenches are distributed in a plausible manner that respects the internal joint constraints and compliances.

Note that the sum of the wrenches at the parent links must equal the wrench applied at the child link k. That is,

$$ {}^{k}w_k = \sum_{i \in \mathcal{P}_k} {}^{i}_{k}\mathsf{Ad}^{T}\ {}^{i}w_i. \tag{15} $$

Consider the case of m superimposed virtual links that was presented in Section 4.3. We can write the following constrained linear system to determine the distribution of wrenches on the different chains:

$$ \begin{bmatrix} \mathsf{C} & \mathrm{I}^T \\ \mathrm{I} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ {}^{k}w_k \end{bmatrix}. \tag{16} $$

The constraint here is the same as Equation 15, except that all quantities are represented in frame k, and thus the adjoints are 6×6 identity matrices, i.e., $\mathrm{I}\ \mathsf{w} = {}^{k}w_k$. To compute a local wrench map that takes the wrench from k and divides it among its parents $1, \ldots, m$, we invert the system above, replacing the right hand side by a block column matrix that will provide the desired vector when right multiplied by ${}^{k}w_k$. That is,

$$ \begin{bmatrix} {}^{1}W_k \\ \vdots \\ {}^{m}W_k \\ * \end{bmatrix} = \begin{bmatrix} \mathsf{C} & \mathrm{I}^T \\ \mathrm{I} & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{17} $$

This gives us a block column vector containing the wrench map for each parent link, with a block $*$ due to the Lagrange multipliers that we can ignore. Note that forming the left hand side blocks ${}^{i}W_k$ requires computing the inverse of a system that may be rank deficient due to the coupled compliance. Again, we use a truncated SVD in its computation.

Finally, while these wrench maps only consider the difficult case of merging (multiple parents) by using superimposed virtual parent links, the transmission of wrenches along simple serial chains is easy. It simply involves a change of coordinates with an adjoint inverse transpose. For parent link a and child link b in a serial chain, the wrench map is simply

$$ {}^{a}W_b = {}^{b}_{a}\mathsf{Ad}^{T}. \tag{18} $$

Algorithm 2 Recursive algorithm to compute all wrench maps ${}^{i}W_e$. Here, i is a mechanism link. Initiate recursion by calling RECURSEWRENCHMAP(e, e, I) for every end effector e, where I is the identity matrix.

```
function RecurseWrenchMap(i, e, iW_e)
 for j ∈ P_i do
  jW_i ← LocalWrenchMap(i)   // apply Equation 17 or 18
  jW_e ← jW_i iW_e           // apply Equation 19
  RecurseWrenchMap(j, e, jW_e)
 end for
end function
```

5.2 Global wrench map

The matrix ${}^{i}W_k$ gives a local mapping for wrench distribution between child link k and parent link i. Since the local wrench map only needs to be computed once, this makes it possible to construct a global wrench map for computing the wrench at internal links due to applied wrenches at the end effectors. Keeping with our scheme of incremental model building, we use the local map to compute the wrenches distributed to internal links due to wrenches applied at the end effectors.

Let ${}^{j}W_i$ be the matrix mapping wrenches from link i to its parent link j. The matrix mapping wrenches from end effector e to link j is simply the compound matrix transform of the wrench map for each body in the path $1, \ldots, n$ between e and j,

$$ {}^{j}W_e = {}^{j}W_1 \left(\prod_{i=1}^{n-1} {}^{i}W_{i+1} \right) {}^{n}W_e. \tag{19} $$

By accumulating the wrenches due to all end effectors, the wrench affecting an internal link is

$$ {}^{i}w_i = \sum_{e \in \mathcal{E}} {}^{i}W_e\, {}^{e}w_e. \tag{20} $$

We use a recursive algorithm to explore the DAG while computing the global wrench map for each link. Algorithm 2 gives an overview of how the equations described in this section are used to build the maps.

5.3 Global twist map

The twist map provides the static solution of the compliant joint chain due to wrenches applied at the end effectors. For a serial chain of compliant joints, the twist at an internal link i is computed as

$$ {}^{i}\phi_i = \sum_{e \in \mathcal{E}} {}^{i}C_{i,i}\ {}^{i}W_e\ {}^{e}w_e. \tag{21} $$

However, for more complex mechanisms, special consideration must be given to the coupled motion due to splitting and merging of the kinematic chain. The contributed motion of links that share a common ancestor with link i must also be considered, and the general version of the twist map ${}^{i}T_e$ in Equation 21 is

$$ {}^{i}T_e = {}^{i}C_{i,i}\ {}^{i}W_e + \sum_{a \in \mathcal{A}} {}^{i}_{a}\mathsf{Ad}\ {}^{a}C_{a,a} \left({}^{a}W_e - {}^{i}_{a}\mathsf{Ad}^{T}\ {}^{i}W_e \right) \tag{22} $$

and ${}^{i}\phi_i = \sum_{e\in\mathcal{E}} {}^{i}T_e\, {}^{e}w_e$. The twist has a component due to the wrench arriving from each ancestor, but also experiences motion due to that of its ancestors influenced by end effector wrenches. The subtraction in the last term ensures that we do not include the motion of the ancestor induced by the wrench transmitted through the chain containing link i, because it is already accounted for in the first term.

6 Dynamic Simulation

We simulate the reduced dynamic system using a backward Euler formulation [31]. As such, we have a system matrix of the form

$$\mathsf{A} = \mathsf{M} - h^2\mathsf{K} - h\mathsf{D}. \tag{23}$$

To solve this system with frictional contacts, we use an iterative projected Gauss-Seidel solver similar to that described by Erleben [32]. This involves a Schur complement of the form $\mathsf{G}^T\mathsf{A}^{-1}\mathsf{G}$, where G is the Jacobian for the contact and friction constraints. We note that it is only necessary to invert the system matrix once, and reuse this small dense inverse system for the duration of the simulation.

The solution to the reduced dynamic system only provides the positions (twists, ξ) and velocities of the end effectors links. Internal links are updated using the twists of the end effectors. Specifically, we compute equivalent static end effector wrenches as $\mathsf{w} = \mathsf{K}\xi$, and from these compute the configuration of internal links using the twist map in Equation 22.

6.1 Free-body base link

For simplicity, the discussion above has let the body frame of the base link be fixed in the world. To extend the reduced model to allow for motion at the base, we integrate a second equation of motion for a rigid body representing the base. We set the base mass and inertia matrix to be that of the entire structure in the rest pose. The motion of the base is driven by gravity, but also by the net external wrenches applied at the end effectors. While the coupled equations of motion could be derived from the Lagrangian, we only couple the base and end-effector models through external forces, and assume that the omitted terms such as Coriolis forces are negligible when the base has large mass and is moving slowly. In this case, the contact and frictional constraints must be modified to use the combined velocity of the base link ${}^{b}\phi_b$ and velocity of the end effector link ${}^{k}\phi_k$ in contact frame c, ${}^{c}\phi_{k+b} = {}^{c}_{k}\mathsf{Ad}\ {}^{k}\phi_k + {}^{c}_{b}\mathsf{Ad}\ {}^{b}\phi_b$.

7 Combining Multiple FORK^{-1}S Models

A single FORK^{-1}S model works well for a moderate range of wrenches applied at the end effectors, but the error in the twist map approximation of the kinematics does grow as the model moves away from the rest post (see Figure 7 in the discussion of results). Furthermore, changes in geometry can bring important changes in the compliance of the end effectors as shown in Figure 4. Therefore, in some scenarios it can be attractive to reduce the kinematic and compliance errors through the use of multiple FORK^{-1}S models at different linearization points. In this section we describe how to blend multiple models so that we can produce accurate simulations even at kinematic states that are disparate from the rest configuration.

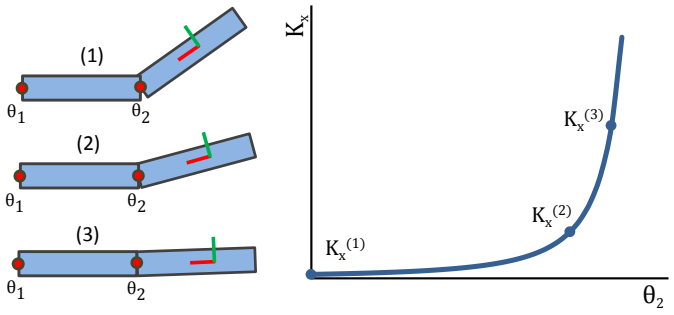

Fig. 4. Three configurations of a two-joint-two-link mechanism shows the need for multiple linearization points. As the right link is pulled horizontally, the compliance in the direction of the local x axis (red) goes to zero as the mechanism approaches a singular configuration. Multiple linearization points provide appropriate stiffnesses for configurations away from the rest pose.

Creating FORK^{-1}S models away from the rest configuration is very similar to the construction of the rest configuration model. We apply a constant set of wrenches at the end effectors to find the static equilibrium solution using a full simulation. With the solution, we construct a new linearized model. Each linearization has a corresponding wrench and reference pose for the end effectors. Therefore, we augment the model to include ${}^{e}\mathsf{b}$, which are the static equilibrium wrenches at the end effectors $e \in \mathcal{E}$, and H_e, which are the homogeneous 4-by-4 transformation matrices giving the configurations of the end effectors at the linearization point.

To combine the models, we use a novel blending scheme that exploits the eigenstructure of the effective mass, stiffness, and damping matrices across multiple models. An estimate of the static equilibrium wrench and configuration is also determined using linear interpolation. We therefore revise the second order dynamical system to include the blended quantities, such that

$$\bar{\mathsf{M}}\dot{\phi} + \bar{\mathsf{D}}\phi + \bar{\mathsf{K}}\xi = \mathsf{w} - \bar{\mathsf{b}}, \tag{24}$$

where $\bar{\mathsf{M}}$, $\bar{\mathsf{D}}$, and $\bar{\mathsf{K}}$ are the blended effective mass, stiffness, and damping matrices, and $\bar{\mathsf{b}}$ is the interpolated static equilibrium wrench. As previously noted, we use a backward Euler formulation of Equation 24, solving for the end effector velocities ϕ.

The final configuration of end effector e is computed as $\bar{H}_e \exp({}^{e}\xi_e)$. Here, $\bar{H}_e$ is the reference configuration of the end effector for the interpolated model. In other words, if the current configuration of the end effectors is the same as one of the sampled configurations, then the matrix $\bar{H}_e$ exactly equals the reference configuration of the closest model; otherwise, it is approximated by interpolating the end effector transformations at each linearization point from the nearby models. We provide additional details on interpolating the displacements and transforms of mechanism bodies in Section 7.3.

7.1 Blending weights

We use a variant of the Shepard interpolation method to compute the blending weights. The weight for the i-th

Algorithm 3 Blend matrices with rank-1 updates. Weights $\alpha_{1\ldots k}$ interpolate the eigendecomposition of matrices $A^{(1\ldots k)}$. Function RANK(A, ϵ) computes the rank of A with tolerance ϵ, and EIG(A) returns the eigendecomposition.

function BLENDRANK1($A^{(1\ldots k)}, \alpha_{1\ldots k}, \epsilon$)
 $\bar{A} \leftarrow 0$
 $r = \text{RANK}(A^{(1)}, \epsilon)$
 $Q^{(1)}, \Lambda^{(1)} \leftarrow \text{EIG}(A^{(1)})$ // Note that this is pre-computed
 for $j \in 1 \ldots r$ **do**
  $\bar{q} \leftarrow \alpha_1 Q^{(1)}_j$
  $\bar{\lambda} \leftarrow \alpha_1 \Lambda^{(1)}_{jj}$
  for $i \in (2 \ldots k)$ **do**
   $Q^{(i)}, \Lambda^{(i)} \leftarrow \text{EIG}(A^{(i)})$ // Note that this is pre-computed
   $q \leftarrow$ vector in $Q^{(i)}_{1\ldots r}$ most similar to $Q^{(1)}_j$
   $\lambda \leftarrow$ diagonal entry in $\Lambda^{(i)}$ corresponding to q
   $\bar{q} \leftarrow q' + \alpha_i q$
   $\bar{\lambda} \leftarrow \bar{\lambda} + \alpha_i \lambda$
  end for
  $\bar{q} \leftarrow \frac{\bar{q}}{\|\bar{q}\|}$
  $\bar{A} \leftarrow \bar{A} + \bar{\lambda}\, \bar{q}\bar{q}^T$
 end for
 return $\bar{A}$
end function

nearby model is

$$\alpha_i = \frac{1}{s}\left(\sum_{e\in E} \|\xi_e - \xi_e^{(i)}\|\right)^{-p}, \tag{25}$$

where s is a normalization factor such that the sum of all α_i equals 1. For the results obtained in this paper, we use $p = 2$. The distance to the nearby model is computed using the current end effector displacement ξ_e and the end effector displacement at the sample configuration $\xi_e^{(i)}$ relative to the base link. We mix linear and angular components of the twist vectors by scaling factors $\frac{1}{r}$ and $\frac{1}{\pi}$ respectively, where r is the bounding sphere radius of the mechanism. These scaling factors are omitted from Equation 25 for brevity.

7.2 Interpolating the effective matrices

The first step in performing simulation by Equation 24 is to compute the interpolated effective matrices. As opposed to simply blending matrix elements, we use the eigenstructure of each matrix so that the effective mass, stiffness, and damping of end effectors can rotate in a smooth manner across the configuration space. Consider the example in Figure 4. As the second joint rotates, the contribution of the compliance of the first joint will be in a different direction in the local coordinate frame. By interpolating the eigenvector directions we can produce a plausible end effector compliance.

We group the eigenvectors of different matrices according to their similarity computed by a dot product. The grouping of eigenvectors across different models is straightforward when the linearized models are not too far from one another in the configuration space. We pre-compute the eigendecomposition of the effective matrices and store the eigenvectors and eigenvalues with each model.

At run time, we interpolate the eigenvectors and corresponding eigenvalues using the blending weights, and then assemble the blended matrix as a sum of rank-1 matrices. These are formed by the outer-products of the interpolated eigenvectors and scaled by interpolated eigenvalues. We preserve the rank of the original matrices. Since the effective matrices are real and symmetric, this is easily done by excluding eigenvalues that have magnitude less than some epsilon value.

The method used to blend effective matrices is provided in Algorithm 3. The eigenstructure of each matrix is pre-computed as Q containing the eigenvectors as columns, and Λ a diagonal matrix containing the eigenvalues. The eigenvalues are sorted by descending magnitude.

As a first step, the rank r is determined by counting the eigenvalues with a magnitude greater than ϵ. The closest model, which corresponds to the largest α_i, is used as a reference and its eigendecomposition is used to determine the rank for the blending. Next, the eigenvalues and eigenvectors are linearly interpolated across all models using the blending weights $\alpha_{1\ldots k}$. Finally, the interpolated eigenvector $\bar{q}$ is normalized and used to performed a rank-1 update to the blended matrix $\bar{A}$, scaled by the interpolated eigenvalue $\bar{\lambda}$.

Algorithm 4 Blending internal links with multiple FORK^{-1}S models using a kinematic approach given end effector twists ξ and blending weights α.

function UPDATEINTERNALLINKS(ξ, $\alpha_{1\ldots k}$)
 ${}^e\mathbf{w} \leftarrow \bar{\mathsf{K}}^e \xi$
 for $l \in \mathcal{L}$ **do** // visit all internal links
  $\bar{\xi}_l \leftarrow 0$
  for $i \in 2\ldots k$ **do**
   $\xi_l^{(i)} \leftarrow \sum_{e\in\mathcal{E}} {}^lT_e^{(i)}\, {}^e w_e^{(i)}$ // use twist map, §5.3
   $\bar{\xi}_l \leftarrow \bar{\xi}_l + \alpha_i \log\left(H_{l\xi}^{(1)^{-1}} H_{l\xi}^{(i)}\right)$ // note $H_{l\xi}^{(i)} = H_l^{(i)}\exp(\xi_l^{(i)})$
  end for
  $\bar{H}_l \leftarrow H_{l\xi}^{(1)} \exp(\bar{\xi}_l)$ // Update simulation body l using $\bar{H}_l$
 end for
end function

7.3 Interpolating the internal body twists

In addition to blending the end effector behavior, the twists of the internal bodies are also interpolated across the models. Rather than using a low-rank blending method involving the twist map and the wrench map, we instead choose to compute the internal wrenches of the blended end effectors and compute the displacements of internal bodies and blend these displacements across all models.

Interpolation of rigid transformations can take inspiration from other work which works with rotations, for instance in averaging quaternion rotations or cubic curves of rotations [33], [34], [35]. To interpolate two models with each providing a twist away from rest, $\xi_j^{(1)}$ and $\xi_j^{(2)}$, we compute the interpolated transform $\bar{H}$ of the end effector as

$$\bar{H} = H_\xi^{(1)} \exp\left(\alpha \log\left(H_\xi^{(1)^{-1}} H_\xi^{(2)}\right)\right) \tag{26}$$

where α is the interpolation parameter and

$$H_\xi^{(j)} = H^{(j)} \exp\left(\xi^{(j)}\right) \tag{27}$$

gives the transform of the end effectoring according to the j-th model.

To include additional models in the interpolation, we use addition with our base model $H^{(1)}\exp(\xi^{(1)})$ serving as

Algorithm 5 The simulation algorithm for blended FORK^{-1}S combines methods for interpolating the effective matrices and displacements of the internal bodies.

function BLENDEDSIMULATION
Compute blending weights $\alpha_{1\ldots k}$ // see Equation 25
$\bar{\mathsf{M}} \leftarrow$ BLENDRANK1($\mathsf{M}^{(1\ldots k)}, \alpha_{1\ldots k}, \epsilon$)
$\bar{\mathsf{K}} \leftarrow$ BLENDRANK1($\mathsf{K}^{(1\ldots k)}, \alpha_{1\ldots k}, \epsilon$)
$\bar{\mathsf{D}} \leftarrow$ BLENDRANK1($\mathsf{D}^{(1\ldots k)}, \alpha_{1\ldots k}, \epsilon$)
for $e \in \mathcal{E}$ **do**
$^e\bar{b}_e \leftarrow \sum_{i=1}^{k} \alpha_i \, {}^e b_e^{(i)}$ // static equilibrium wrench
$\bar{H}_e \leftarrow H_e^{(1)} \exp\big(\sum_{i=2}^{k} \alpha_i \log({H_e^{(1)}}^{-1} H_e^{(i)})\big)$ // ref. config.
end for
$\xi \leftarrow$ STEPENDEFFECTORDYNAMICS // solve Equation 24
UPDATEINTERNALLINKS(ξ, $\alpha_{1\ldots k}$) // see Algorithm 4
end function

the linearization point for other models. The observation is that this works well as long as the error for both our base model and the other models are close enough. The other observation is that while order is important when chaining multiplications, by doing interpolation with addition inside the exponential we avoid the order dependence of matrix multiplication. Specifically,

$$\bar{H} = H^{(1)} \exp\Big(\sum_{j=2}^{k} \alpha_j \log({H^{(1)}}^{-1} H^{(j)})\Big). \qquad (28)$$

Algorithm 4 outlines the method used to perform kinematic blending of the internal links. Much as in the single model approach, the end effector wrenches $^e\mathbf{w}$ are estimated as the end effector displacements $^e\xi$ transformed by the interpolated stiffness. For each model, the displacement contributed by each end effector is computed using the twist map $^l T_e^{(i)}$, which is specific to the link and the model. This is then used to update the interpolated displacement $\bar{\xi}_l$ of link l using Equation 28. Note that our implementation of the Algorithm 4 is parallelized, per link, across the models.

7.4 Blended FORK^{-1}S simulation

The procedure for dynamical simulation of the blended model is outlined in Algorithm 5. The first step is to determine the blending weights based on the current displacements of the effectors. We then compute the interpolated mass, stiffness, and damping matrices, along with the static equilibrium wrench and reference configuration for each end effector. With the blended model, we then step the simulation of the end effectors and update positions of internal links.

7.5 Model selection

Non-parametric regression methods, such as the k-nearest neighbors (k-NN) algorithm, seem well-suited for our blending approach. However, this requires selecting a subset of the sampled models and interpolating across them. We have observed that excluding models from blending can produce popping artifacts in the motion of the internal links. Therefore, we choose to perform blending across all of the sampled models. While blending across all models is wasteful, we recognize that a good avenue of future research would be to group models into well-behaved subsets and to build a graph of these subsets that can be walked at runtime.

In some instances, one or more of the effectives matrices in the blending set are lower rank than the others. This typically occurs if the mechanism configuration is singular when the model is constructed. In other words, the actual degrees of freedom of the system is not reflected by the rank of the effective matrices. For example, the undeformed configuration of the ladder in Figure 5 is singular since it cannot be stretched or deformed in a vertical direction; that is, the vertical direction is inadmissible. This causes problems with blending, since we do not want to introduce inadmissible displacements to the simulation of the mechanism.

One solution is to blend only eigenvectors that correspond to admissible twist directions across all models, and exclude directions contained only in the higher rank models. As the simulation moves away from a singular configuration, the reduced rank matrix can be dropped and the additional admissible directions blended in. Another option is to artificially increase the rank of the lower rank matrices and blend them with the others. For the stiffness matrix, this corresponds to blending in a very large stiffness; for mass, a near infinite mass. Equivalently, in our examples we take the approach of blending the inverse effective matrices, and zero is used for inadmissible directions when the set contains lower rank matrices.

8 RESULTS

We have integrated our approach with the Vortex simulator of CMLabs Simulations. In this section we demonstrate the utility of our approach and show different models that we can produce for various scenarios of importance. Examples include simulation of multi-legged robot in contact with uneven terrain, and simulation of human grasping, as seen in Figure 1 and in the accompanying video. Note that all video results were obtained by FORK^{-1}S simulation, unless otherwise stated.

Figure 5 shows a comparison between simulations with the single model FORK^{-1}S method versus a commercial physics engine. In each case, a constant force is applied at an end effector and simulated until static equilibrium is reached. The final configurations are perceptually very similar between simulation with FORK^{-1}S and the full body physics engine.

8.1 Blended mechanisms

Figure 6 shows a comparison of mechanisms simulated using single model FORK^{-1}S versus the blended approach described in Section 7. The single model method begins to display constraint violation errors when the end effectors are displaced away from the initial configuration. This is because a single linearization point is no longer a good approximation of the dynamical or kinematic behavior of the mechanism. By including models at additional linearization points, the non-linear behavior of the mechanism is better represented by smoothly blending in variations of end effector dynamics.

The accompanying video demonstrates that new models can be added quickly and interactively to the blending

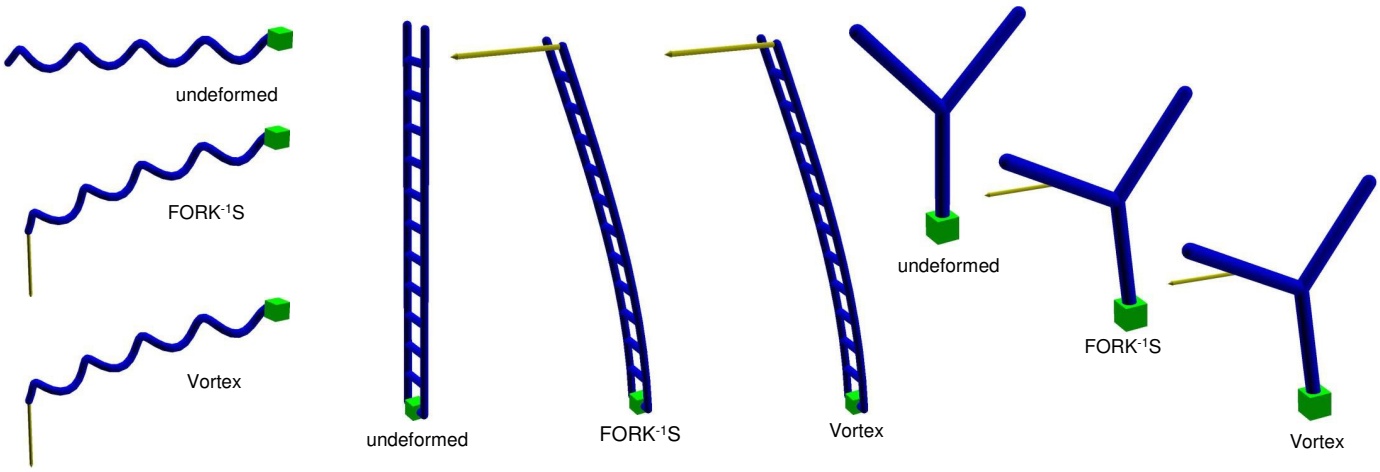

Fig. 5. Deforming a helix (left), spring ladder (middle), and "Y" mechanism (right). The rest configuration is shown, as well as a comparison between the static solution reach by simulation with FORK-1S vs. a constrained rigid body simulator (Vortex). In each case, a 100 N force is applied (yellow arrow); all joints use a compliance of $C_\theta = 0.001$.

set. Forces are applied at an end effector of the ladder mechanism using a virtual spring that pulls the end effector toward the on-screen cursor. When constraint violation errors are visually perceptible using the FORK-1S simulation, the user switches to using a full constrained multi-body simulation. A new model is constructed once static equilibrium is reached. The eigenvalue decomposition of the effective matrices of the new model are computed and stored for blending, which is not a computationally expensive step given that the matrices are as small as 6-by-6 if there is only one end-effector. Finally, switching back to the FORK-1S simulation, the new model is blended with the other models.

For the results in this paper, we manually select the linearization points using this interactive approach. However, we conceptualize that an automatic method could be devised whereby the selection process is guided by the constraint violation and a maximum error threshold. For instance, by randomly deforming the mechanism across its state space.

Example	# links	Vortex	FORK-1S			Speed-up
			N=1	N=4	N=8	
Helix	50	240	20	12	15	20.0×
Helix	100	470	32	16	21	29.4×
Helix	400	2150	112	84	76	28.3×
Ladder	48	334	22	14	18	23.9×
Robot arm	20	121	24	18	21	6.7×

TABLE 1
The mean computation time in μs per simulation step for various examples. Our method with N threads is compared against performing a full constrained rigid body simulation using the Vortex physics engine.

Example	# links	Vortex	# of FORK-1S			Speed-up
			1	2	4	
Helix	200	1048	78	96	160	13.4×
Ladder	36	302	37	82	99	8.2×
Robot arm	20	121	26	47	86	4.7×

TABLE 2
The mean computation time in μs per simulation step for various examples using the blended FORK-1S method. In each case, 4 threads were used to simulate the blended model.

8.2 Constraint violation error

We perform a number of experiments to explore, quantitatively, how well the FORK-1S method approximates the behavior of a mechanism. Specifically, we measure constraint violation error versus end effector displacement. The error at each joint is measured as the Euclidean distance between constraint attachment points of body pairs.

Figure 7 shows the relative constraint error for the "Y" and the ladder mechanism (shown in Figure 5) as an end effector is displaced by applying an external force in constant direction and slowly increasing the magnitude. The relative error is computed as the position violation of each constraint, accumulated over all joints and scaled by $\frac{1}{r}$, where r is the bounding sphere radius of the mechanism at the rest configuration. This is plotted versus the relative displacement of the end effector, which is computed as the Euclidean norm of the twist with the linear component scaled by $\frac{1}{r}$ and the angular component scaled by $\frac{1}{\pi}$.

We note that the rate of increase in the error is dependent on the direction of the applied force, but even for significant displacements of the end-effector, the proportional constraint error remains low. For example, when the relative end effector displacement is 1, the single model FORK-1S method has a relative violation error in the range of 0.1 to 0.15. However, by using the blended FORK-1S method, the error for a larger range of displacements is significantly reduced. From Figure 7 it is clear that the blended FORK-1S method interpolates the sample points (i.e., the error becomes zero) in addition to reducing the overall constraint violation error.

8.3 Performance

Here, we compare the overhead of simulating a mechanism with a constrained rigid body physics engine versus the method outlined in this paper. An Intel Core i7 2.8 GHz processor with 4 cores was used to obtain performance results.

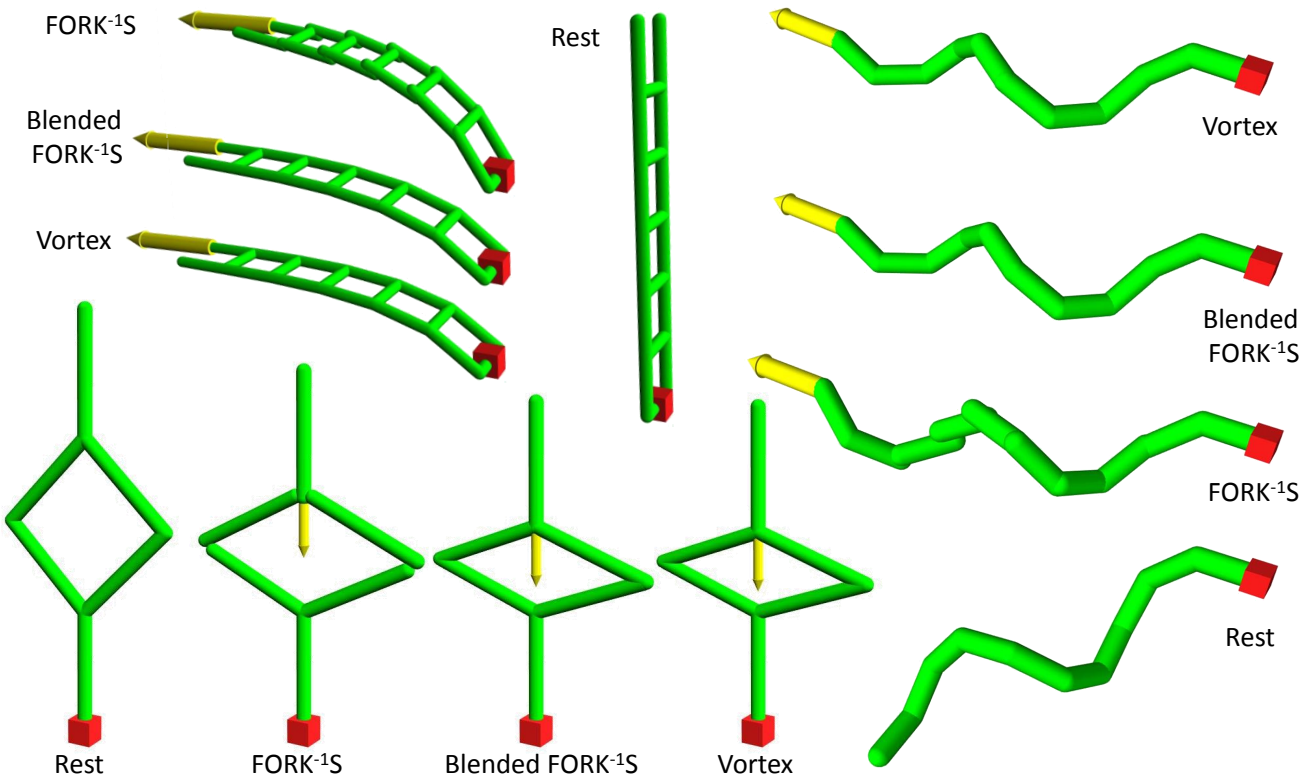

Fig. 6. Deforming a ladder (upper left), helix (right), and loop structures (bottom left), comparing blended FORK-1S to single model FORK-1S. A force is applied at the center of mass of end effector bodies with a direction indicated by the yellow arrow. In the case of blending, only two models were used and the static equilibrium configuration shown in these examples was not included as a linearization point.

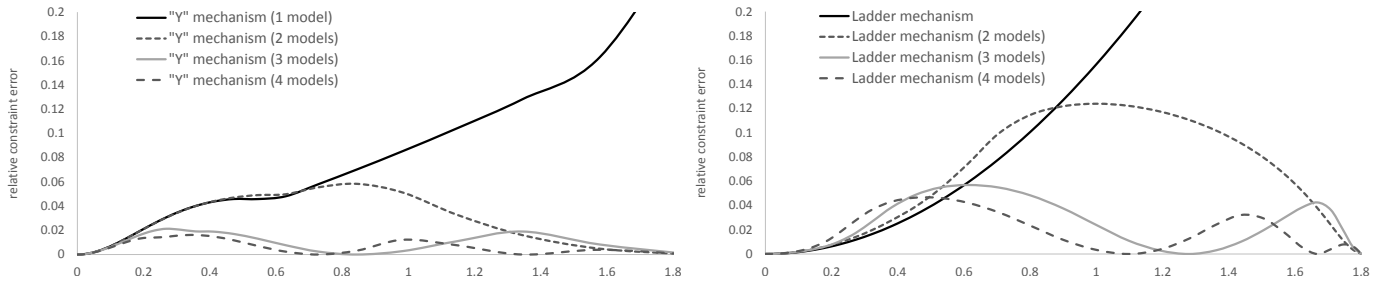

Fig. 7. The relative constraint error is measured for the "Y" shaped mechanism (left) and ladder mechanism (right) as a single end effector is pulled is various directions. The vertical axis gives the constraint violation error, which is relative to the mechanism size. The horizontal axis gives the relative displacement of the end effector, which is also represented in proportion to the mechanism.

Note that only moderate efforts were made to optimize our implementation.

The computation times for solving the dynamical system in Section 6 and performing numerical integration are given in Table 1. Each mechanism listed in the table was simulated using a single threaded version of the Vortex physics engine, as well as our single model FORK-1S implementation using different numbers of threads for the parallel update of internal links.

The FORK-1S method performed better in all cases, with the most drastic speedups observed when simulating long serial chains. Notably, there is a 28 times performance increase for the helix example with 400 links. Also, as Table 1 suggests, there is a sweet spot in choosing the thread count for simulating a particular mechanism, with an increase of threads not necessarily giving better performance. One practical consideration that impacts performance significantly is grouping the internal links so that updates are performed in batches per thread. This avoid unnecessary context switching. Additionally, the associated data structures are stored contiguously in memory in order to minimize memory thrashing issues.

Table 2 shows the computation time per step for simulating various mechanisms using the blended FORK-1S approach. We note that the computation time increases with the number of models. This is not unexpected since the data structures of individual models must be updated in addition to computing the parameters of the blended model. Specifically, our implementation is integrated with the Vortex physics engine and there is overhead due to conversion between data structures used by FORK-1S. However, the blended simulation remains a notable improvement over the

full constrained rigid body simulation. Also, Equation 26 is used to update the reference configuration of the end effectors as well as the configuration of the internal links. Conversion of the homogeneous transform matrix using the exponential map becomes a bottle neck is this case, and further performance improvement could be realized by optimizing our implementation of this function.

8.4 Discussion and limitations

Interior body reconstructions have constraint errors when large external forces are applied at the end effectors of a simulation with just a single FORK^{-1}S model. Therefore one of our contributions with this work is to introduce models at additional linearization points and blend their behavior as the system is pulled from its rest state.

Although the added models extend the range of motion for a mechanism, geometric limits of the internal joints are not necessarily well approximated by our method. Adaptively stiffening the system may help in these situations, become stiffer as singular and joint limit configurations are approached. This strategy could also be used to avoid states where joint constraint errors appear. We leave joint limits and additional non-linear compliance scaling for future work.

It is also possible to make small modifications to the geometry to correct errors at the expense of letting rigid links deform, and such strategies have been used in repairing foot skate [36] and length changes are often not perceived [37]. Errors can be fixed by allowing rigid bodies to stretch, but there is a limit to how large external forces can grow before geometry modifications are visible.

We note that the behavior of the reduced model can differ from the full model. In general, we observe the reduced systems to be slightly stiffer than their fully simulated systems. This is not surprising, and we believe this occurs naturally due to the lower number of degrees of freedom and the linearization we impose. Higher levels of damping seen in the reduced system can be explained by our implicit integration, while Vortex uses a symplectic integration scheme.

The construction process assumes that we can walk from a base node in the graph to all end effectors. When there are loop closures between two end effectors that are on the far side of the graph from the base, the incremental algorithm will not find them. An alternative projection technique is necessary in this case.

9 Conclusion

First order reduced models of compliant mechanisms provide a fast alternative to simulating virtual humans and robots. By focusing only on the end effectors, the simulation only needs to solve a small dense system while the full state of the non-reduced mechanism can be computed in parallel. Using the exponential map to compute the state of the internal links produces a desirable behavior with little separation at joints for a good range of interaction forces. Our method deals with loops in the constraints, and permits different levels of physics fidelity by adjusting the number of end effectors included in the reduced model. Finally, with the method we present for blending multiple linearization points we can accurately simulate structures over a wider range of simulation states.

FORK^{-1}S provide an important new approach among a large spectrum of techniques important for the creation of interactive and immerse virtual environments. We believe it will be a useful tool for improving industrial training simulations and physics-based character animation.

9.1 Future Work

A number of avenues of future work are discussed in Section 8.4. We also intend to investigate various methods for controlling end effector motion. This is useful for many applications, such as manipulation tasks in character animation and robotics simulation. Online control for dexterous manipulation tasks has been successfully demonstrated using PD servo control [38]. We believe a similar control framework could be achieved using FORK^{-1}S, for example, by modulating the static equilibrium wrench to drive the gripper towards a set point posture rather than interpolating the value of b_e across the models. Similarly, our reduced compliant model could be used to perform short-horizon motion planning using model predictive control (MPC), for instance, in combination with the technique described by Kumar et al. [39]. We believe that reduced models for MPC are an important avenue of future research.

Appendix - Rigid Body Kinematics

Any rigid motion from one position to another may be described as a *screw* motion. That is, there exists a coordinate frame in which the motion consists of a translation along an axis combined with a rotation about the same axis. The time derivative of a screw motion is a *twist* consisting of the linear velocity $v \in \mathbb{R}^3$ and angular velocity $\omega \in \mathbb{R}^3$. Since much of this paper concerns statics, and because it is convenient to write rigid displacements (screws) in body coordinate frames, we abuse the term *twist* for these small displacements ξ. We use ϕ and the term *velocity* to write the equations of motion and specifically use the *body velocity* as defined by Murray et al. [28]. Analogous to a twist, a *wrench* $w \in \mathbb{R}^6$ is a generalized force consisting of a linear force $f \in \mathbb{R}^3$ and a rotational torque $\tau \in \mathbb{R}^3$. Following Murray et al., we pack twist and wrench vectors with linear parts on top and angular parts on the bottom, i.e., $\phi = (v^T \omega^T)^T$ and $w = (f^T \tau^T)^T$.

Twists and wrenches transform to different coordinate frames using the adjoint matrix $\mathsf{Ad} \in \mathbb{R}^{6\times 6}$. To transform twists from coordinate frame a to coordinate frame b, we directly use

$$ {}^b_a\mathsf{Ad} = \begin{bmatrix} {}^b_aR & {}^b\hat{p}_a\, {}^b_aR \\ 0 & {}^b_aR \end{bmatrix}, \tag{29} $$

where ${}^b_aR \in SO(3)$ is the rotation matrix from frame a to b, the origin of coordinate frame a in coordinates of frame b is ${}^bp_a$, and $\hat{}$ is the cross product operator. That is, ${}^a\phi$ in frame b is computed as ${}^b\phi = {}^b_a\mathsf{Ad}\,{}^a\phi$. The inverse transpose of the adjoint is used to transform a wrench between coordinate frames, ${}^bw = {}^b_a\mathsf{Ad}^{-T}\,{}^aw$. Finally, we use the exponential map $e^{\hat{\phi}} : \mathbb{R}^6 \to SE(3)$ on a twist to compute the relative rigid motion as a homogeneous transformation

matrix using formulas given by Murray et al. [28]. Here, we follow Murray et al. where the hat notation $\hat{\phi}$ denotes a repacking of the 6 components of a twist into a 4-by-4 matrix. However, throughout this paper we omit the hat notation and let it be implicitly clear from context that the exponential of a twist is indeed the matrix exponential.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their suggestions for improving the paper. This work was supported by funding from NSERC, CFI, MITACS, CINQ, and GRAND NCE.

REFERENCES

[1] S. Andrews, M. Teichmann, and P. G. Kry, "FORK-1S: Interactive compliant mechanisms with parallel state computation," in *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, ser. I3D '14, 2014, pp. 7–14. [Online]. Available: http://doi.acm.org/10.1145/2556700.2556717

[2] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006.

[3] M. Nesme, P. G. Kry, L. Jeřábková, and F. Faure, "Preserving topology and elasticity for embedded deformable models," *ACM Trans. on Graphics*, vol. 28, no. 3, p. 52, 2009.

[4] J. Barbič and Y. Zhao, "Real-time large-deformation substructuring," *ACM Trans. on Graphics*, vol. 30, no. 4, pp. 91:1–91:8, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964986

[5] T. Kim and D. James, "Physics-based character skinning using multidomain subspace deformations," *IEEE Trans. on Visualization and Computer Graphics*, vol. 18, no. 8, pp. 1228–1240, 2012.

[6] D. Harmon and D. Zorin, "Subspace integration with local deformations," *ACM Trans. on Graphics*, vol. 32, no. 4, pp. 107:1–107:10, 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461922

[7] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Trans. on Visualization and Computer Graphics*, vol. 12, no. 1, pp. 36–47, 2006.

[8] E. G. Parker and J. F. O'Brien, "Real-time deformation and fracture in a game environment," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 2009, pp. 165–175. [Online]. Available: http://doi.acm.org/10.1145/1599470.1599492

[9] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *Proc. of IEEE International Conference on Robotics and Automation*, 2006, pp. 1904–1909.

[10] J. Allard, F. Faure, H. Courtecuisse, F. Falipou, C. Duriez, and P. G. Kry, "Volume contact constraints at arbitrary resolution," *ACM Trans. on Graphics*, vol. 29, no. 4, p. 82, 2010.

[11] C. O'Sullivan and J. Dingliana, "Collisions and perception," *ACM Trans. on Graphics*, vol. 20, no. 3, pp. 151–168, 2001. [Online]. Available: http://doi.acm.org/10.1145/501786.501788

[12] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 137–146. [Online]. Available: http://doi.acm.org/10.1145/237170.237226

[13] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, 1st ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1998.

[14] U. Ascher and P. Lin, "Sequential regularization methods for simulating mechanical systems with many closed loops," *SIAM Journal on Scientific Computing*, vol. 21, no. 4, pp. 1244–1262, 1999.

[15] F. Faure, "Fast iterative refinement of articulated solid dynamics," *IEEE Trans. on Visualization and Computer Graphics*, vol. 5, no. 3, pp. 268–276, 1999.

[16] R. Tomcin, D. Sibbing, and L. Kobbelt, "Efficient enforcement of hard articulation constraints in the presence of closed loops and contacts," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 235–244.

[17] R. Featherstone and D. Orin, "Robot dynamics: equations and algorithms," *Proc. of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 826–834, 2000.

[18] R. Featherstone, *Rigid Body Dynamics Algorithms*. New York: Springer, 2008.

[19] R. M. Mukherjee and K. S. Anderson, "A logarithmic complexity divide-and-conquer algorithm for multi-flexible articulated body dynamics," *Journal of Computational and Nonlinear Dynamics*, vol. 2, no. 1, pp. 10–21, 2006.

[20] S. Redon, N. Galoppo, and M. C. Lin, "Adaptive dynamics of articulated bodies," *ACM Trans. on Graphics*, vol. 24, no. 3, pp. 936–945, 2005.

[21] P. G. Kry, L. Reveret, F. Faure, and M. P. Cani, "Modal locomotion: Animating virtual characters with natural vibrations," *Computer Graphics Forum*, vol. 28, no. 2, pp. 289–298, 2009.

[22] R. F. Nunes, J. B. Cavalcante-Neto, C. A. Vidal, P. G. Kry, and V. B. Zordan, "Using natural vibrations to guide control for locomotion," in *Proceedings of the ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games*, ser. I3D '12. New York, NY, USA: ACM, 2012, pp. 87–94. [Online]. Available: http://doi.acm.org/10.1145/2159616.2159631

[23] K. Yamane and Y. Nakamura, "Natural motion animation through constraining and deconstraining at will," *IEEE Trans. on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 352–360, 2003.

[24] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics Tools*, vol. 10, 2004.

[25] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[26] A. Z. Hajian and R. D. Howe, "Identification of the mechanical impedance at the human finger tip," *Journal of biomechanical engineering*, vol. 119, no. 1, pp. 109–114, 1997.

[27] C. J. Hasser and M. R. Cutkosky, "System identification of the human hand grasping a haptic knob," in *Proc. of the 10th Symp. on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, 2002. [Online]. Available: http://ieeexplore.ieee.org/iel5/7836/21555/00998957.pdf

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

[29] P. Hansen, "Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 3, pp. 503–518, 1990. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/0911028

[30] W. T. V. W. H. Press, S. A. Teukolsky and B. P. Flannery, *Numerical Recipes (3rd edition)*. Cambridge University Press, 2007.

[31] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proc. of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 43–54.

[32] K. Erleben, "Velocity-based shock propagation for multibody dynamics animation," *ACM Trans. on Graphics*, vol. 26, no. 2, 2007. [Online]. Available: http://doi.acm.org/10.1145/1243980.1243986

[33] M.-J. Kim, M.-S. Kim, and S. Y. Shin, "A general construction scheme for unit quaternion curves with simple high order derivatives," in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95, 1995, pp. 369–376. [Online]. Available: http://doi.acm.org/10.1145/218380.218486

[34] S. R. Buss and J. P. Fillmore, "Spherical averages and applications to spherical splines and interpolation," *ACM Trans. Graph.*, vol. 20, no. 2, pp. 95–126, Apr. 2001. [Online]. Available: http://doi.acm.org/10.1145/502122.502124

[35] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.

[36] L. Kovar, J. Schreiner, and M. Gleicher, "Footskate cleanup for motion capture editing," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 2002, pp. 97–104. [Online]. Available: http://doi.acm.org/10.1145/545261.545277

[37] J. Harrison, R. A. Rensink, and M. van de Panne, "Obscuring length changes during animated motion," *ACM Trans. on Graphics*, vol. 23, no. 3, pp. 569–573, 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015761

[38] S. Andrews and P. G. Kry, "Goal directed multi-finger manipulation: Control policies and analysis," *Computers & Graphics*, vol. 37, no. 7, pp. 830–839, 2013.

[39] V. Kumar, Y. Tassa, T. Erez, and E. Todorov, "Real-time behaviour synthesis for dynamic hand-manipulation," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6808–6815.

Sheldon Andrews received a B.Eng. degree in computer engineering from Memorial University, St. John's, Canada in 2004, a M.A.Sc. degree in electrical engineering from the University of Ottawa, Canada in 2007, and a Ph.D. in computer science from McGill University, Montreal, Canada in 2014. He worked as a software developer from 2007 to 2009 implementing and designing real-time physics simulations at CMLabs Simulations in Montreal, Canada. He is currently a postdoctoral researcher at Disney Research in Edinburgh, UK. His research interests include human motion synthesis, physics-based animation, grasping and dexterous manipulation, measurement based modeling for virtual environments, and intelligent systems. Dr. Andrews is a member of the IEEE.

Marek Teichmann completed his Ph.D. in computer science at the Courant Institute, NYU, in the field of Computational Geometry and Robotics, working on theoretical and practical aspects of grasping and fixturing. Marek was collision group leader at Lateral Logic, a developer of visualization and simulation software technology. Marek continued this work at MathEngine, where he designed and implemented advanced collision algorithms as part of computer simulations of rigid-body dynamics systems. He is currently CTO of CMLabs Simulations.

Paul G. Kry received his B.Math. in computer science with electrical engineering electives in 1997 from the University of Waterloo, and his M.Sc. and Ph.D. in computer science from the University of British Columbia in 2000 and 2005. He spent time as a visitor at Rutgers during most of his Ph.D., and did postdoctoral work at INRIA Rhne Alpes and the LNRS at Universit Ren Descartes. He is currently an associate professor at McGill University. His research interests are in physically based animation, including deformation, contact, motion editing, and simulated control of locomotion, grasping, and balance. He co-chaired ACM/EG Symposium on Computer Animation in 2012, Graphics Interface in 2014, and served on numerous program committees, including ACM SIGGRAPH, ACM/EG Symposium on Computer Animation, Pacific Graphics, and Graphics Interface. He heads the Computer Animation and Interaction Capture Laboratory at McGill University. Paul Kry is currently the president of the Canadian Human Computer Communications Society, the organization which sponsors the annual Graphics Interface conference.