FORK^{-1}S: Interactive compliant mechanisms with parallel state computation

Sheldon Andrews[*] McGill University
Marek Teichmann[†] CM Labs Simulations
Paul G. Kry[‡] McGill University

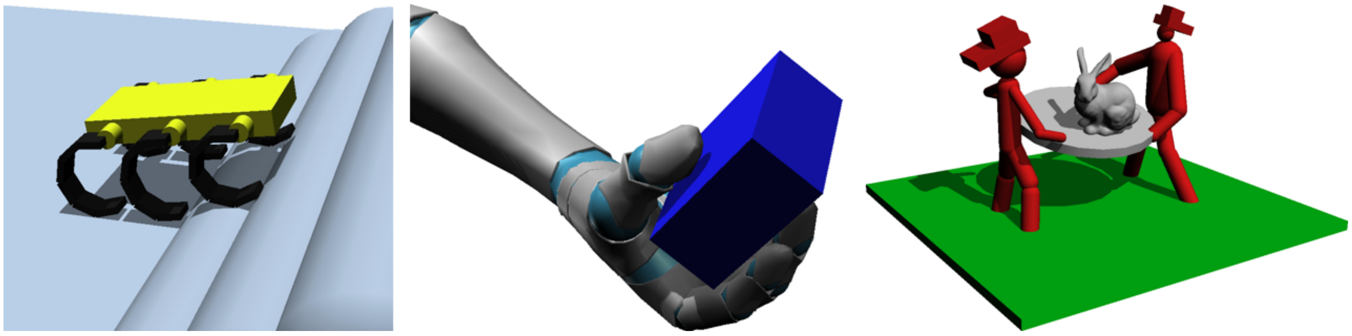

Figure 1: *Example simulations that we use to demonstrate our technique include a multi-legged robot on uneven terrain, two firemen catching a bunny, and humanoid robot grasping. The reduced model for the firemen involves only one body, the trampoline, while the reduced models for the legged robot and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

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. 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.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: character animation, physics simulation, constraints

1 Introduction

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

In this paper, we describe a technique that can improve interactive simulations of scenarios 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 interactive 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 our articulated systems experience contact. Thus, we can 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 for computing the dynamics model, which walks the body connectivity graph. 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.

[*]e-mail:sheldon.andrews@mail.mcgill.ca
[†]e-mail:marek@cm-labs.com
[‡]e-mail:kry@cs.mcgill.ca

At simulation time, external forces produce a dynamic transient behavior for the bodies that we include in the model. The position of all the remaining bodies is 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 and 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 adequate range of interaction forces, and we discuss the size of the errors produced. 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 from smart phones to desktop computers 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.

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.

We demonstrate various important 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, for instance, in 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. [2006], while other alternative elastic simulation reduction techniques continue to be an active area of research [Nesme et al. 2009; Barbič and Zhao 2011; Kim and James 2012; Harmon and Zorin 2013]. 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 [Duriez et al. 2006; Parker and O'Brien 2009; Yamane and Nakamura 2006]. With respect to the contact equations, the contact patches can be discretized at arbitrary resolutions [Allard et al. 2010]. Finally, collision detection and response can be modified to produce various plausible animations with different fidelity levels [O'Sullivan and Dingliana 2001].

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 [Baraff 1996]. Constrained multi-body simulations are popular for their simplicity, and available in a number of different libraries including Vortex, PhysX, Havok, Box2D, and the Open Dynamics Engine. Numerical drift must be addressed in this case using stabilization techniques [Ascher and Petzold 1998], and loops result in redundant constraints that require additional attention in the solution of such systems [Ascher and Lin 1999; Faure 1999].

The alternative to full-coordinates is to formulate the system in minimal coordinates, i.e., the joint angles [Featherstone and Orin 2000]. Straightforward linear time solvers have been used for decades, while divide and conquer approaches permit parallel algorithms with log time complexity [Featherstone 2008; Mukherjee and Anderson 2006]. 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 RTQL8 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 resembles a coupled elastic mass-spring system. 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 [Redon et al. 2005]. In contrast, with a constrained redundant coordinate formulation, modal reduction of rigid articulated structures is possible and has been used to animate animal locomotion [Kry et al. 2009].

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 [Yamane and Nakamura 2003] or damped least squares [Buss and Kim 2004]. 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.

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 [1987]. 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 [Hajian and Howe 1997; Hasser and Cutkosky 2002]. 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. [1994] 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.

3 Dynamics projection and notation

Our method targets mechanisms made up of articulate 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 will 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

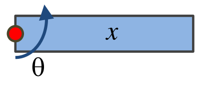

Figure 2: *A single rigid body link rotating about a compliant joint.*

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

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

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

$$K_\theta^{-1}\tau = \Delta\theta, \tag{2}$$

where K_θ^{-1} is the compliance, or inverse stiffness, of the joint.

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

$$JK_\theta^{-1}J^T w = \xi. \tag{4}$$

Here, $K_x^{-1} = JK_\theta^{-1}J^T$ 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

$$K_x^{-1}w_{ext} = \xi \tag{5}$$

and the homogeneous transformation of the body's displacement is computed by the exponential map.

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, K_x^{-1} is 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) [Hansen 1990] to compute the inverse when needed.

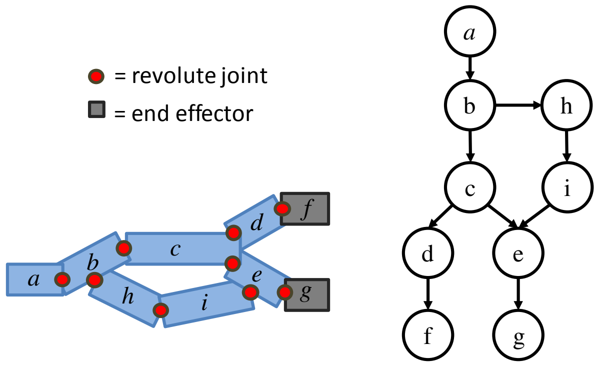

Figure 3: *An image visualizing the connectivity of bodies used in constructing the effective coupled stiffness, damping, and mass of end effectors.*

We can perform a very similar projection of the rotational mass matrix M_θ. From the equation of motion $M_\theta^{-1}\tau = \ddot{\theta}$, and assuming the body acceleration $\dot{\phi} \in \mathbb{R}^6$ is approximately $J\ddot{\theta}$, we find $JM_\theta^{-1}J^T w = \dot{\phi}$, thus, $M_x^{-1} = JM_\theta^{-1}J^T$. We follow the same projection for the damping matrix to produce the second order system

$$M_x\dot{\phi} + D_x\phi + K_x\xi = w. \tag{6}$$

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 6 has 6 dimensions while it was constructed from a 1D joint. Nevertheless, 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 Notation

Throughout the rest of the paper, we will be using 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. We use P_i and C_i to denote the set of parents and children of link i. Also, A_j is the set of ancestors of link j where branching occurs, e.g., from Figure 3, A_f contains c and b. Finally, let E denote the set of end effectors.

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

Procedure 1 Recursive algorithm for computation of K^{-1}. Initiate recursion with RECURSECOMPLIANCE(*base*).

```
function RECURSECOMPLIANCE($i$)
    if $|$PARENTS($i$)$| == 1$ then
        $K_{i,i}^{-1} \leftarrow$ CHAIN($i$)                // apply Equation 7
    else if $|$PARENTS($i$)$| > 1$ then
        $K_{i,i}^{-1} \leftarrow$MERGE($i$)                 // apply Equation 11
    end if
    for $j \in$ CHILDREN($i$) do
        RECURSECOMPLIANCE($j$)
    end for
    if $|$CHILDREN($i$)$| > 1$ then
        SPLIT($i$)                                   // apply Equation 9
    end if
end function
```

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 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 due to the stiffness of the joint between the two bodies, K_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 k. The twist-wrench relationship at link k is

$$K_{b,b}^{-1} = J K_\theta^{-1} J^T + {}^{b}_{a}\mathsf{Ad}\; K_{a,a}^{-1}\; {}^{b}_{a}\mathsf{Ad}^T, \tag{7}$$

where $K_{a,a}^{-1}$ is the effective compliance of the parent link, J and K_θ^{-1} are respectively the kinematic Jacobian and compliance of the common joint. Multiplying the compliance $K_{b,b}^{-1}$ 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 $K_{b,b}^{-1}$ 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 ${}^{b}_{b}K_{b,b}^{-1}$ 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 $\Phi = \mathsf{K}^{-1}\mathsf{w}$ where

$$\mathsf{K}^{-1} = \begin{bmatrix} K_{1,1}^{-1} & \cdots & K_{1,m}^{-1} \\ \vdots & \ddots & \vdots \\ K_{m,1}^{-1} & \cdots & K_{m,m}^{-1} \end{bmatrix} \tag{8}$$

and $\Phi = ({}^{1}\phi_1^T \cdots {}^{m}\phi_m^T)^T$, $\mathsf{w} = ({}^{1}w_1^T \cdots {}^{m}w_m^T)^T$. In block matrix K^{-1} the diagonal block $K_{i,i}^{-1}$ is the compliance of child link i computed as described for the chaining case, while the off-diagonal blocks provide the coupling. That is, $K_{i,j}^{-1}$ determines the twist at link i due to a wrench applied to 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:

$$K_{i,j}^{-1} = {}^{i}_{k}\mathsf{Ad}\; K_{k,k}^{-1}\; {}^{j}_{k}\mathsf{Ad}^T. \tag{9}$$

Chaining additional links after a split is very similar. The diagonal blocks are updated as per Equation 7, while the off diagonal blocks are updated using Equation 9, 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 $1, \ldots, m$, and coupled compliance matrix K^{-1} computed with Equation 8. Let us now write the linear system describing the wrench-twist relationship using the stiffness as

$$\mathsf{w} = \mathsf{K}\Phi \tag{10}$$

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, ${}^{k}w_k = \sum_{i=1\ldots m} {}^{k}w_i$. Therefore, the effective stiffness at link k is

$$K_{k,k} = \mathrm{I}\;\mathsf{K}\;\mathrm{I}^T \tag{11}$$

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. As an aside, performing matrix inversion by a truncated SVD is used extensively in our work. The tolerance parameter for inclusion of singular values is tuned according to the stiffnesses of mechanism joints, and this is done by inspection.

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 Procedure 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,

$$^{k}w_k = \sum_{i \in P_k} {}^{i}_{k}\mathsf{Ad}^T \, {}^{i}w_i. \tag{12}$$

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:

$$\begin{bmatrix} \mathsf{K}^{-1} & I^T \\ \mathrm{I} & 0 \end{bmatrix} \begin{bmatrix} \mathsf{w} \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ {}^{k}w_k \end{bmatrix}. \tag{13}$$

The constraint here is the same as Equation 12, 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,

$$\begin{bmatrix} {}^{1}W_k \\ \vdots \\ {}^{m}W_k \\ * \end{bmatrix} = \begin{bmatrix} \mathsf{K}^{-1} & I^T \\ \mathrm{I} & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix}. \tag{14}$$

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 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

$$^{a}W_b = {}^{b}_{a}\mathsf{Ad}^T. \tag{15}$$

Procedure 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($i, e, {}^{i}W_e$)
    for $j \in$ Parents($i$) do
        ${}^{j}W_i \leftarrow$ LocalWrenchMap($i$) // apply Equation 14 or 15
        ${}^{j}W_e \leftarrow {}^{j}W_i {}^{i}W_e$          // apply Equation 16
        RecurseWrenchMap($j, e, {}^{j}W_e$)
    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,

$$^{j}W_e = {}^{j}W_1 \left(\prod_{i=1}^{n-1} {}^{i}W_{i+1} \right) {}^{n}W_e. \tag{16}$$

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

$$^{i}w_i = \sum_{e \in E} {}^{i}W_e \, {}^{e}w_e. \tag{17}$$

We use a recursive algorithm to explore the DAG while computing the global wrench map for each link. Procedure 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

$$^{i}\phi_i = \sum_{e \in E} {}^{i}K_{i,i}^{-1} \, {}^{i}W_e \, {}^{e}w_e. \tag{18}$$

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 in Equation 18 is

$$\sum_{e \in E} \left({}^{i}K_{i,i}^{-1} \, {}^{i}W_e + \sum_{a \in A} {}^{i}_{a}\mathsf{Ad} \, K_{a,a}^{-1} \left({}^{a}W_e - {}^{i}_{a}\mathsf{Ad}^T \, {}^{i}W_e \right) \right). \tag{19}$$

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 (Equation 18).

6 Dynamic simulation

We simulate the reduced dynamic system using a backward Euler formulation [Baraff and Witkin 1998]. As such, we have a system matrix of the form $\mathsf{A} = \mathsf{M} - h^2\mathsf{K} - h\mathsf{D}$. To solve this system with frictional contacts, we use an iterative projected Gauss-Seidel solver similar to that described by Erleben [2007]. 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 then from these, compute the configuration of internal links using the twist map in Equation 19.

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 Results

We have integrated our approach with the Vortex simulator of CM Labs. In this section we demonstrate the utility of our approach and the models we can produce in various scenarios of importance, such as simulation of amphibious robot walking, 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 4 shows a comparison between simulations with our method versus a commercial physics engine. In each case, a constant force is applied at an end effector and simulated until a static equilibrium is reached. The final configurations are perceptually very similar between simulation with FORK^{-1}S and the full body physics engine.

Additionally, we have performed a number of experiments to explore how joint constraint violation errors grow as external forces are applied. The error at each joint is measured as the Euclidean distance between constraint attachment points of the body pairs, which are accumulated over all joints and scaled by the bounding sphere radius of the mechanism.

Example	# links	Vortex	FORK^{-1}S		
			N=1	N=4	N=8
Helix	50	240	20	12	15
Helix	100	470	32	16	21
Helix	400	2150	112	84	76
Ladder	48	334	22	14	18
Robot arm	20	121	24	18	21

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.*

Figure 5 shows error plots for the "Y" mechanism and robotic arm as a single end effector is pulled in various directions. The relative error is computed as the position violation of the constraint, accumulated over all joints and scaled by the bounding sphere radius r of the mechanism. 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 r and the angular component scaled by 2π. 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. Perceptually there is often no constraint violation, as demonstrated in the accompanying video. However, although the error remains relatively low, our method is based on a low order approximation, and once the mechanism is sufficiently displaced from its initial configuration the error increases sharply.

7.1 Performance

Here, we compare the overhead of simulating a mechanism with a constrained rigid body physics engine versus the method outlined in this paper. The computation times for solving the dynamical system in Section 6 and performing numerical integration is 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 FORK^{-1}S implementation using different numbers of threads for the parallel update of internal links. Note that only moderate efforts were made to optimize our implementation.

The FORK^{-1}S 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 400 link helix example. 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.

7.2 Discussion and limitations

Constraint errors result in our interior body reconstruction when we apply large external forces. As previously discussed, these 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. As such, we have imagined the addition of a model to reduce compliance as the system is pulled from its rest state. This would help address the fact that the structure should become stiffer as singular configurations are approached, and would help us approximately model geometric limits of the internal joints. We plan to implement such a non-linear compliance scaling method in future work.

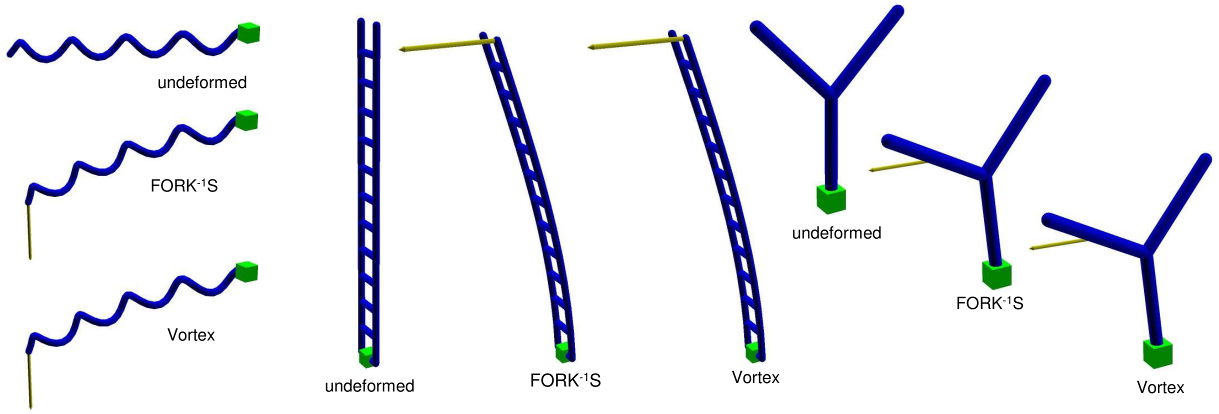

Figure 4: *Deforming a helix (left), spring ladder (middle), and "Y" mechanism (right). The rest configuration is shown, as well as a comparison between the static solution reach by simulation with FORK^{-1}S vs. a constrained rigid body simulator (Vortex). In each case, a* $100N$ *force is applied (yellow arrow);all joints use a stiffness of* $K = 1000$.

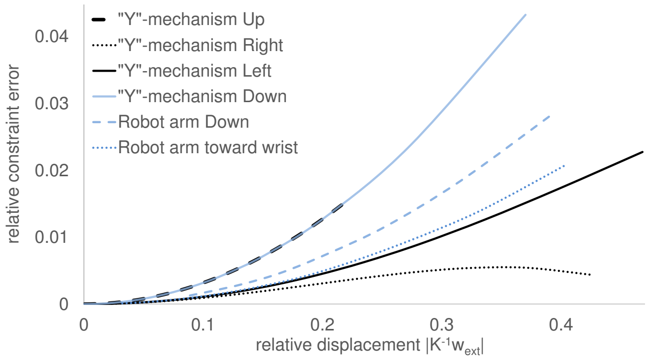

Figure 5: *The relative constraint error is measured for the "Y" shaped mechanism and the robotic arm as a single end effector is pulled is various directions. The vertical axis gives the constraint violation error, which is relative to the mechanism size. The horizontal axis gives the relative displacement of the end effector, which is also represented in proportion to the mechanism.*

Adaptively stiffening the system would help keep the state closer to the rest configuration, avoiding states where joint constraint violations would be visible. Nevertheless, we also note that it is possible to make small modifications to the geometry to correct the error at the expense of letting rigid links deform. Such a strategy has been used in repairing foot skate [Kovar et al. 2002], and such length changes are often not perceived [Harrison et al. 2004].

We note that the behavior of our 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.

Finally, 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.

8 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 good 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. FORK^{-1}S provide an important new approach among a large spectrum of techniques important for the creation of interactive and immerse virtual environments, and we believe it will be a useful tool in improving training simulations once fully integrated into Vortex simulation software.

8.1 Future Work

A number of avenues of future work are discussed in Section 7.2. We also intend to investigate various methods for controlling end effector motion. This is useful for many applications, such as manipulation tasks for character animation and robotics simulation. Rather than assuming a static joint configuration with PD servo control, it may be possible to linearize the compliant behavior at multiple configurations across pose space. By blending across these models, the compliant behavior of the mechanism could be simulated for the entire motion trajectory. Additionally, for applications that require gain scheduling, the state space for the linearization can include the combined joint pose and stiffness parameters.

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 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. [1994]. 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

$$ {}^b_a\mathsf{Ad} = \begin{bmatrix} {}^b_aR & {}^b\hat{p}_a\,{}^b_aR \\ 0 & {}^b_aR \end{bmatrix}, \tag{20} $$

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. [1994].

Acknowledgements

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

ALLARD, J., FAURE, F., COURTECUISSE, H., FALIPOU, F., DURIEZ, C., AND KRY, P. G. 2010. Volume contact constraints at arbitrary resolution. *ACM Trans. on Graphics 29*, 4, 82.

ASCHER, U., AND LIN, P. 1999. Sequential regularization methods for simulating mechanical systems with many closed loops. *SIAM Journal on Scientific Computing 21*, 4, 1244–1262.

ASCHER, U. M., AND PETZOLD, L. R. 1998. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, 1st ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of the 25th annual conference on Computer graphics and interactive techniques*, 43–54.

BARAFF, D. 1996. Linear-time dynamics using lagrange multipliers. In *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, 137–146.

BARBIČ, J., AND ZHAO, Y. 2011. Real-time large-deformation substructuring. *ACM Trans. on Graphics 30*, 4 (July), 91:1–91:8.

BUSS, S. R., AND KIM, J.-S. 2004. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools 10*.

DURIEZ, C., DUBOIS, F., KHEDDAR, A., AND ANDRIOT, C. 2006. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Trans. on Visualization and Computer Graphics 12*, 1, 36–47.

ERLEBEN, K. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. on Graphics 26*, 2.

FAURE, F. 1999. Fast iterative refinement of articulated solid dynamics. *IEEE Trans. on Visualization and Computer Graphics 5*, 3, 268–276.

FEATHERSTONE, R., AND ORIN, D. 2000. Robot dynamics: equations and algorithms. *Proc. of IEEE International Conference on Robotics and Automation 1*, 826–834.

FEATHERSTONE, R. 2008. *Rigid Body Dynamics Algorithms*. Springer, New York.

HAJIAN, A. Z., AND HOWE, R. D. 1997. Identification of the mechanical impedance at the human finger tip. *Journal of biomechanical engineering 119*, 1, 109–114.

HANSEN, P. 1990. Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. *SIAM Journal on Scientific and Statistical Computing 11*, 3, 503–518.

HARMON, D., AND ZORIN, D. 2013. Subspace integration with local deformations. *ACM Trans. on Graphics 32*, 4, 107:1–107:10.

HARRISON, J., RENSINK, R. A., AND VAN DE PANNE, M. 2004. Obscuring length changes during animated motion. *ACM Trans. on Graphics 23*, 3, 569–573.

HASSER, C. J., AND CUTKOSKY, M. R. 2002. 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*.

KHATIB, O. 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation 3*, 1, 43–53.

KIM, T., AND JAMES, D. 2012. Physics-based character skinning using multidomain subspace deformations. *IEEE Trans. on Visualization and Computer Graphics 18*, 8, 1228–1240.

KOVAR, L., SCHREINER, J., AND GLEICHER, M. 2002. Footskate cleanup for motion capture editing. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 97–104.

KRY, P. G., REVERET, L., FAURE, F., AND CANI, M. P. 2009. Modal locomotion: Animating virtual characters with natural vibrations. *Computer Graphics Forum 28*, 2, 289–298.

MUKHERJEE, R. M., AND ANDERSON, K. S. 2006. A logarithmic complexity divide-and-conquer algorithm for multi-flexible articulated body dynamics. *Journal of Computational and Nonlinear Dynamics 2*, 1, 10–21.

MURRAY, R., LI, Z., AND SASTRY, S. S. 1994. *A mathematical introduction to robotic manipulation*. CRC Press.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4, 809–836.

NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. on Graphics 28*, 3, 52.

O'SULLIVAN, C., AND DINGLIANA, J. 2001. Collisions and perception. *ACM Trans. on Graphics 20*, 3, 151–168.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 165–175.

REDON, S., GALOPPO, N., AND LIN, M. C. 2005. Adaptive dynamics of articulated bodies. *ACM Trans. on Graphics 24*, 3, 936–945.

YAMANE, K., AND NAKAMURA, Y. 2003. Natural motion animation through constraining and deconstraining at will. *IEEE Trans. on Visualization and Computer Graphics 9*, 3, 352–360.

YAMANE, K., AND NAKAMURA, Y. 2006. Stable penalty-based model of frictional contacts. In *Proc. of IEEE International Conference on Robotics and Automation*, 1904–1909.