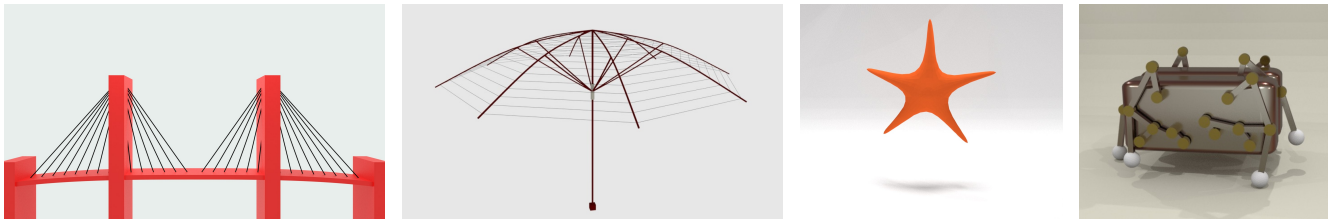
RedMax: Efficient & Flexible Approach for Articulated Dynamics

YING WANG, Texas A&M University
NICHOLAS J. WEIDNER, Texas A&M University
MARGARET A. BAXTER, Texas A&M University
YURA HWANG, Texas A&M University
DANNY M. KAUFMAN, Adobe Research
SHINJIRO SUEDA, Texas A&M University



(a) (b) (c) (d)

Fig. 1. (a) Near linear time evaluation for a cable-stayed Bridge. (b) Near linear time evaluation for a deployable Umbrella. (c) Starfish, showing fully two-way coupled integration between articulated and deformable bodies. (d) Klann walker, with rapid evaluation of internal friction within joints.

It is well known that the dynamics of articulated rigid bodies can be solved in $O(n)$ time using a recursive method, where n is the number of joints. However, when elasticity is added between the bodies (*e.g.,* damped springs), with linearly implicit integration, the stiffness matrix in the equations of motion breaks the tree topology of the system, making the recursive $O(n)$ method inapplicable. In such cases, the only alternative has been to form and solve the system matrix, which takes $O(n^3)$ time. We propose a new approach that is capable of solving the linearly implicit equations of motion in near linear time. Our method, which we call RedMax, is built using a combined reduced/maximal coordinate formulation. This hybrid model enables direct flexibility to apply arbitrary combinations of constraints and contact modeling in both reduced and maximal coordinates, as well as mixtures of implicit and explicit forces in either coordinate representation. We highlight RedMax's flexibility with seamless integration of deformable objects with two-way coupling, at a standard additional cost. We further highlight its flexibility by constructing an efficient internal (joint) and external (environment) frictional contact solver that can leverage bilateral joint constraints for rapid evaluation of frictional articulated dynamics.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Physical simulation, Rigid body dynamics, Constraints, Contact, Friction

Authors' addresses: Ying Wang, Texas A&M University; Nicholas J. Weidner, Texas A&M University; Margaret A. Baxter, Texas A&M University; Yura Hwang, Texas A&M University; Danny M. Kaufman, Adobe Research; Shinjiro Sueda, Texas A&M University.

0730-0301/2019/7-ART104 $15.00
https://doi.org/10.1145/3306346.3322952

ACM Reference Format:
Ying Wang, Nicholas J. Weidner, Margaret A. Baxter, Yura Hwang, Danny M. Kaufman, and Shinjiro Sueda. 2019. RedMax: Efficient & Flexible Approach for Articulated Dynamics. *ACM Trans. Graph.* 38, 4, Article 104 (July 2019), 10 pages. https://doi.org/10.1145/3306346.3322952

1 INTRODUCTION

Articulated rigid body dynamics has many applications in various disciplines, including biomechanics, robotics, aerospace, and computer graphics. It has been extensively studied starting in the 1960s (*e.g.,* [Roberson 1966]), but it was not until the 1980s that an $O(n)$ algorithm, where n is the number of joints or bodies, became widely known [Featherstone 1983]. This algorithm and its variants are based on a recursive formulation, where various quantities are computed recursively based on the tree structure of the mechanism. Around the same time, an alternative $O(n^3)$ method based on matrix factorization was also developed [Walker and Orin 1982], which, according to De Jalon and Bayo [2012], can outperform the $O(n)$ recursive method when n is small (< 10). Although some important mechanisms, such as serial manipulators, have only a few joints, for many applications in computer graphics, n can be quite large—even a single hand has $n \geq 15$. Therefore, there are still many cases where $O(n)$ methods are still preferred over $O(n^3)$ methods.

The story changes when implicit elasticity is added between *arbitrary* bodies rather than only between immediate neighbors. Examples of such scenarios include: simply attaching damped springs between pairs of bodies; architectural design with cables [Whiting et al. 2012; Deuss et al. 2014]; musculoskeletal simulations with line-based forces [Delp et al. 2007; Wang et al. 2012]; and deployable folding mechanisms [Demaine and O'Rourke 2008; Zhou et al. 2014]. Using the linearly implicit integrator commonly used in graphics [Baraff and Witkin 1998], the $O(n)$ recursive method no longer

works because the stiffness matrix breaks the tree topology of the system matrix. To date, the only alternative has been to use the $O(n^3)$ factorization method. We address this issue by introducing a new approach that allows us to solve the linearly implicit equations of motion in linear to subquadratic time. If the topology-breaking springs are not present, our method gracefully reverts back to the standard $O(n)$ recursive approach.

In the discussion so far, we have tacitly assumed that the dynamics are represented using "reduced" coordinates, where a minimal set of degrees of freedom (DOFs), such as joint angles for revolute joints and relative translations for prismatic joints, are used to represent the state of the system. An alternate approach that uses "maximal" coordinates has also been studied.[1] For example, an $O(n)$ method for maximal coordinates was discovered by Baraff [1996]. However, constraints need to be applied to model joints, and these constraints must be stabilized to avoid drift [Baumgarte 1972; Cline and Pai 2003]. On the other hand, reduced coordinates do not require any stabilization, since reduced coordinates only allow configurations that satisfy the joint constraints. Loops are handled with constraints in either approach, but in practice, stabilizing a few loop constraints is much easier than stabilizing the whole structure. Furthermore, reduced coordinates are in general faster, because the number of DOFs is much smaller (*e.g.,* 1/6 the size), and no constraints are required. Also, Baraff [1996] notes that there is anecdotal evidence that larger time steps are possible using reduced coordinates.

One of the advantages of maximal coordinates is that it is more intuitive—it is easier to add various implicit/explicit forces and to combine with other deformable objects. To address this point, we show that our formulation of dynamics, which we call RedMax, is very flexible and extensible. Any combination of reduced/maximal forces can be added implicitly or explicitly, and any combination of reduced/maximal constraints can be handled. Furthermore, it becomes trivial to get full two-way coupling between a deformable object (such as an FEM) and the articulated rigid bodies.

As a further demonstration of the flexibility of our formulation, we show that we can also incorporate frictional contact within the joints in an efficient manner, by taking advantage of the reduced coordinate representation of the joints. Our approach works well when augmented with both bilateral (*e.g.,* loop closure) and unilateral (*e.g.,* external contact) constraints.

To summarize, our contributions are:

- §3: A near linear approach for articulated dynamics, even in the presence of the maximal stiffness matrix, based on our novel matrix-free Projected Block Jacobi Preconditioner.
- §4: A formulation that exposes both maximal and reduced degrees of freedom, allowing any combination of implicit and explicit forces and constraints in either coordinates. It also handles full, implicit two-way coupling between articulated and deformable bodies.
- §5: Frictional dynamics for jointed mechanisms that takes advantage of the bilateral nature of the joint constraints. Our approach works seamlessly with other constraints, including loop-closing constraints and external collision constraints.

[1]Maximal coords are also called absolute coords or Cartesian coords; reduced coords are also called generalized coords or minimal coords.

2 RELATED WORK

Articulated rigid body dynamics has been an active research area for many decades, especially in the field of robotics, where high-performance algorithms were required for low-power systems. A great deal of effort has been spent on both $O(n)$ and $O(n^3)$ methods, for example, to refine the performance for tree-configuration or closed-loop systems [Bae and Haug 1987a,b]. Although asymptotically worse, $O(n^3)$ methods have attracted significant attention because they are more intuitive and are more easily parallelizable [Walker and Orin 1982; Avello et al. 1993; Negrut et al. 1997]. However, these methods do not work in the presence of the maximal stiffness matrix from linearly implicit integration, one of the most common integration methods in graphics [Baraff and Witkin 1998].

The use of reduced/maximal coordinates with two way coupling of articulated and deformable bodies has been of particular interest in computer graphics. Shinar et al. [2008] used maximal coordinate dynamics with pre-stabilization, coupled with a finite element mesh. Their method achieves full two-way coupling, but their intricate time-stepping method makes it difficult to extend or to incorporate into existing simulators. Kim and Pollard [2011] used reduced coordinate dynamics with explicit coupling between rigid and deformable bodies. Their method is extremely fast to evaluate; however their approach is limited to explicit integration schemes, since the reduced coordinates are integrated with the $O(n)$ recursive method. Jain and Liu [2011] used reduced coordinates with fully implicit coupling between rigid and deformable bodies. However, with their formulation, each deformable body can only be influenced by a single rigid body. Liu et al. [2013] used an off-the-shelf solver for maximal coordinate articulated bodies, which was time-stepped alongside a deformable body. Their coupling, however, was limited to explicit interactions—they only affected each other after taking a time step. To summarize, our method is unique in that it is capable of simultaneously using reduced coordinates, with fully implicit two-way coupling, and with the deformable mesh spanning multiple rigid bodies. We should clarify, however, that we limited our discussion to two-way coupling between articulated and deformable bodies—the works above provide many other important contributions.

There have also been a number of related works on the simulation of various phenomena using articulated rigid bodies, such as: trees [Quigley et al. 2018], hair [Hadap 2006], and characters [Hernandez et al. 2011]. Linear time methods for *flexible* multibody systems have also been studied for decades, as described in the detailed survey by Wasfy and Noor [2003]. Of particular importance to graphics, Bertails [2009] showed that the recursive linear time approach can be used to simulate the dynamics of elastic rods. These efficient methods can only be used in the special case when all of the implicit forces are between topologically neighboring bodies (*e.g.,* joint springs), since then the topology of the reduced stiffness matrix will be the same as that of the reduced mass matrix. However, in the general case, the implicit forces are between *arbitrary* bodies, and so the recursive linear time approaches cannot be used.

To model joint friction for articulated bodies formulated in reduced coordinates, a common approach is to treat the frictional force solely as a function of joint velocities, rather than using the geometry of the joint [Drumwright and Shell 2010; Sciavicco and

Table 1. Table of notation. Eq. (S2.5) refers to suppl. doc. §2, Eq. (5).

$\mathsf{q}_r, \dot{\mathsf{q}}_r, \ddot{\mathsf{q}}_r$	Reduced coords, velocity, and acceleration
$\mathsf{q}_m, \dot{\mathsf{q}}_m, \ddot{\mathsf{q}}_m$	Maximal coords, velocity, and acceleration
$\mathsf{J}_{mr}, \dot{\mathsf{J}}_{mr}$	Jacobian from $\dot{\mathsf{q}}_r$ to $\dot{\mathsf{q}}_m$, and its time derivative
$\mathsf{M}_r, \mathsf{f}_r$	Reduced mass matrix and force vector
$\mathsf{M}_m, \mathsf{f}_m$	Maximal mass matrix and force vector
$\tilde{\mathsf{M}}_r, \tilde{\mathsf{f}}_r$	The LHS matrix and RHS vector of Eq. (2)
$\mathsf{K}_r, \mathsf{D}_r$	Reduced stiffness and damping matrices
$\mathsf{K}_m, \mathsf{D}_m$	Maximal stiffness and damping matrices
$\dot{\mathsf{q}}^{(k)}, \dot{\mathsf{q}}^{(k+1)}$	Velocity at step k or $k+1$ (reduced or maximal)
${}^a_b\mathsf{Ad}$	Adjoint transform from b to a (Eq. (S2.5))
S	Joint Jacobian (Eq. (S3.14))
$\mathsf{G}_r, \mathsf{G}_m$	Reduced & maximal bilateral constraint matrices
$\mathsf{C}_r, \mathsf{C}_m$	Reduced & maximal unilateral constraint matrices
$\mathsf{M}_d, \mathsf{f}_d, \dot{\mathsf{q}}_d$	Deformable mass, force, velocity
$\dot{\mathsf{q}}^{\text{prev}}$	Last velocity (same as $\dot{\mathsf{q}}^{(k)}$); used in §5
N, T	Contact normal and tangent matrices
α, β	Contact and friction Lagrange multipliers
$\mathsf{f}_\alpha, \mathsf{f}_\beta$	Contact and friction forces (maximal)

Siciliano 2012]. With our approach, we use the geometry of the joint in our friction algorithm, and we show that changing the geometry parameters affects the resulting motion. Finally, some recent research has shown the effectiveness of using a non-discretized friction cone [Acary and Brogliato 2008; Li et al. 2018]. However, we note that the joints in most mechanisms have only 1 DOF, and so the friction "cone" can be trivially modeled by a box constraint.

3 PROJECTED BLOCK JACOBI PRECONDITIONER

Recursive O(n) methods for the forward and inverse dynamics of articulated mechanisms, with support for both reduced and maximal coordinates have existed for decades [Popov et al. 1978; Featherstone 1983; Baraff 1996; Negrut et al. 1997; Serban et al. 1997]. Unfortunately, when there are maximal springs applying implicit forces on the rigid bodies (*e.g.,* Figs. 1a & 1b), we can no longer use these recursive $O(n)$ methods, because the stiffness matrix resulting from these springs breaks the tree topology of the system matrix. In this section, we define our RedMax approach (Eq. (2)), and we introduce a new preconditioner that gives linear to subquadratic time performance, depending on the scene, even in the presence of the maximal stiffness matrix (Eq. (3)).

We assume that the reader is familiar with maximal and reduced coordinates. For those who need a refresher, we include short tutorials in the supplemental document §2 & §3, respectively. We also include a table of notation in Table 1.

We start with the reduced equations of motion (derived in §3 of the supplemental document, Eq. (S3.11)):

$$\left(\mathsf{J}_{mr}^\top \mathsf{M}_m \mathsf{J}_{mr}\right) \ddot{\mathsf{q}}_r = \mathsf{J}_{mr}^\top \left(\mathsf{f}_m - \mathsf{M}_m \dot{\mathsf{J}}_{mr}\, \dot{\mathsf{q}}_r\right), \tag{1}$$

where M_m is the diagonal maximal mass matrix, q_r is the reduced configuration (*e.g.,* joint angles), f_m is the maximal force, and J_{mr} is the Jacobian for transforming from reduced velocity to maximal velocity: $\dot{\mathsf{q}}_m = \mathsf{J}_{mr}\dot{\mathsf{q}}_r$. The Jacobian and its time derivative, $\mathsf{J}_{mr}, \dot{\mathsf{J}}_{mr}$, are of size $\#m \times \#r$, where $\#m$ is the number of maximal DOFs, and $\#r$ is the number of reduced DOFs. Eq. (1) is an instance of the well-known "velocity transformations" for articulated dynamics [De Jalon and Bayo 2012], with our SE(3) based Jacobian, J_{mr}. This equation of motion can be used in conjunction with different choices of time integrators.

We now describe our RedMax formulation, which is a particular discretization of Eq. (1) that exposes all the reduced and maximal quantities. We follow the common practice in graphics and discretize Eq. (1) at the velocity level. Then combining the linearly implicit terms for both reduced and maximal coordinates [Baraff and Witkin 1998], we arrive at our RedMax formulation:

$$\begin{aligned}\left(\mathsf{J}_{mr}^\top \left(\mathsf{M}_m + h\mathsf{D}_m - h^2\mathsf{K}_m\right)\mathsf{J}_{mr} + h\mathsf{D}_r - h^2\mathsf{K}_r\right)\dot{\mathsf{q}}_r^{(k+1)} = \\ \left(\mathsf{J}_{mr}^\top \mathsf{M}_m \mathsf{J}_{mr}\right)\dot{\mathsf{q}}_r^{(k)} + h\left(\mathsf{f}_r + \mathsf{J}_{mr}^\top\left(\mathsf{f}_m - \mathsf{M}_m \dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r^{(k)}\right)\right),\end{aligned} \tag{2}$$

where D_m and K_m are the maximal damping and stiffness matrices, D_r and K_r are the reduced damping and stiffness matrices, f_r is the reduced force vector, f_m is the maximal force vector, including the Coriolis force, and $\dot{\mathsf{q}}_r^{(k)}$ and $\dot{\mathsf{q}}_r^{(k+1)}$ are the reduced velocities at time steps k and $k+1$. The last term, $-\mathsf{J}_{mr}^\top \mathsf{M}_m \dot{\mathsf{J}}_{mr}\,\dot{\mathsf{q}}_r$, is the extra *quadratic velocity vector* due to the change of coordinates [Shabana 2013]. This long equation gives us the flexibility to choose the types of forces we want to use, be they maximal, reduced, explicit, or implicit. These terms may come from a variety of sources, including geometric stiffness [Tournier et al. 2015]. For example, we can easily combine body damping (D_m), maximal springs acting on the bodies (K_m and f_m), joint damping (D_r), and joint stiffness (K_r and f_r). We will highlight more of RedMax's flexibility in §4 and §5. For brevity, when appropriate, we will use the shorthand notation $\tilde{\mathsf{M}}_r\dot{\mathsf{q}}_r = \tilde{\mathsf{f}}_r$ instead of Eq. (2).

An important fact about Eq. (2) is that, because the maximal stiffness matrix breaks the tree-structure of the system, it cannot be solved by the $O(n)$ recursive dynamics algorithm. (For reference, we include the recursive dynamics algorithm as a supplemental document.) In the rest of this section, we introduce our preconditioner that gives linear to subquadratic performance in the presence of the maximal stiffness matrix.

Before we delve into the details of our preconditioner, we first clarify when it works best. Our preconditioner is effective when the rigid DOFs make up a large portion of the system DOFs, and when these rigid DOFs are tied together by *maximal* forces, such as damped springs between various bodies. These cover some important simulation scenarios, including architectural design with cables [Whiting et al. 2012; Deuss et al. 2014] and biomechanical simulations with line-based forces [Delp et al. 2007; Wang et al. 2012]. When there are no maximal springs, our preconditioner still gives the same performance as the $O(n)$ recursive approach—it gracefully reverts back to the standard $O(n)$ approach. When a deformable object with many DOFs is being simulated simulateously, we instead use a standard direct solver.

Our preconditioner, P, can be expressed as follows:

$$\mathsf{P} = \mathsf{J}_{mr}^\top(\mathsf{M}_m + \text{blkdiag}(h\mathsf{D}_m - h^2\mathsf{K}_m))\mathsf{J}_{mr} + h\mathsf{D}_r - h^2\mathsf{K}_r, \tag{3}$$

where 'blkdiag' is a filter that keeps only the 6×6 diagonal blocks of D_m and K_m. We call this the Projected Block Jacobi Preconditioner because we take the block diagonals of the maximal terms and project them into the reduced space. Using P, we solve the

Algorithm 1 Computes $y = (\mathsf{M}_r + \mathsf{J}_{mr}^\top \text{blkdiag}(h\mathsf{D}_m - h^2\mathsf{K}_m)\mathsf{J}_{mr} + h\mathsf{D}_r - h^2\mathsf{K}_r)^{-1}x$ in linear time for preconditioning a linearly implicit solver. Script j refers to the current joint, i to the associated body, c to the joint's child joint, and p to the joint's parent joint.

1: // Run this loop once as a preprocessing step
2: **while** backward traversal **do** ▷ c = child joint of j
3: $\mathsf{A}_j^m = {}_j^i\mathsf{Ad}^\top \text{blkdiag}(h\mathsf{D}_i^m - h^2\,\mathsf{K}_i^m)\,{}_j^i\mathsf{Ad}$ ▷ Maximal term
4: $\mathsf{A}_j^r = h\mathsf{D}_j^r - h^2\mathsf{K}_j^r$ ▷ Reduced term
5: $\hat{\mathsf{M}}_j = (\mathsf{M}_j + \mathsf{A}_j^m) + \sum_c {}_j^c\mathsf{Ad}^\top\, \Pi_c\, {}_j^c\mathsf{Ad}$
6: $\Psi_j = (S_j^\top \hat{\mathsf{M}}_j S_j + \mathsf{A}_j^r)^{-1}$
7: $\Pi_j = \hat{\mathsf{M}}_j - \hat{\mathsf{M}}_j S_j \Psi_j S_j^\top \hat{\mathsf{M}}_j$
8: **end while**
9:
10: // Run these two loops for each RHS vector x
11: **while** backward traversal **do** ▷ c = child joint of j
12: $\hat{\mathcal{B}}_j = \sum_c {}_j^c\mathsf{Ad}^\top \beta_c$
13: $\beta_j = \hat{\mathcal{B}}_j + \hat{\mathsf{M}}_j(S_j\Psi_j(x_j - S_j^\top\hat{\mathcal{B}}_j))$
14: **end while**
15: **while** forward traversal **do** ▷ p = parent joint of j
16: $y_j = \Psi_j(x_j - S_j^\top \hat{\mathsf{M}}_j\, {}_p^j\mathsf{Ad}\,\dot{\mathsf{V}}_p - S_j^\top\hat{\mathcal{B}}_j)$
17: $\dot{\mathsf{V}}_j = {}_p^j\mathsf{Ad}\,\dot{\mathsf{V}}_p + S_j y_j$
18: **end while**

preconditioned linear system $\mathsf{P}^{-1}\tilde{\mathsf{M}}_r\dot{\mathsf{q}}_r = \mathsf{P}^{-1}\tilde{\mathsf{f}}_r$ in a *matrix-free* fashion. To use P in the preconditioned conjugate gradient (PCG) method to solve Eq. (2) in near linear time, we have the following requirements:

(1) Form the RHS vector of Eq. (2) in $O(n)$ time.
(2) Multiply a vector by the LHS matrix of Eq. (2) in $O(n)$ time.
(3) Apply the preconditioner, P, in $O(n)$ time.
(4) Converge in a sublinear number of iterations.

For (1) and (2), we must be able to multiply a vector by J, $\mathsf{J}^\top$, and $\dot{\mathsf{J}}$, as required by the RHS of Eq. (2), in $O(n)$ time. Although filling these matrices takes $O(n^2)$ time, computing the product can be done in $O(n)$ time, by taking advantage of the recursive nature of the topology (see Supplemental §3). To multiply by J and $\dot{\mathsf{J}}$, we traverse forward starting from the root, whereas to multiply by $\mathsf{J}^\top$, we traverse backward starting from the leaf. The recursive dynamics method takes this approach, while computing the reduced velocities and forces. Adding the spring contributions to the RHS force and LHS stiffness matrix can be done trivially in $O(m)$ time using standard techniques, where m is the number of springs.

To enable (3), we must be able to *solve* by P in linear time. We draw inspiration from the fact that the recursive *forward* dynamics algorithm solves the reduced system $\mathsf{M}_r\ddot{\mathsf{q}}_r = \mathsf{f}_r$ in linear time, allowing it be utilized to construct, or to multiply by, the inverse inertia matrix efficiently, by setting all forces and velocities to zero [Kim 2012; Drumwright 2012]. In the same way, our preconditioner can be used to solve the block diagonal approximation of the LHS matrix of Eq. (2) in linear time. We add two important modifications to the standard recursive forward dynamics algorithm, corresponding to the maximal and reduced implicit terms, as shown lines 3-6 in Alg. 1. These two types of implicit terms must be handled differently, since they operate in different spaces. In each joint j, we store the reduced stiffness and damping matrices (scalars for revolute joints), and in the corresponding body i, we store the 6×6 block diagonal components of the maximal stiffness and damping matrices. The reduced terms are added prior to taking the inverse, in line 6 of Alg. 1. The maximal terms are first transformed to be in j's coordinate space and then are added to the j's inertia matrix in line 5. These terms are then processed recursively together with the inertia.

For (4), we offer empirical evidence based on the scaling of the Bridge (Fig. 1a) and the Umbrella (Fig. 1b) scenes in §6, which show sublinear number of iterations per time step. In the Bridge scene, when the towers are infinitely stiff, PCG converges in 1 iteration, because all of the cables are attached to a stationary body, and so the stiffness matrix becomes block diagonal.

3.1 Loop Closure

Our preconditioner is applicable also when there are constraints, of which loop-closure is the most common. For example, in the Bridge scene shown in Fig. 1a, we apply bilateral loop-closing constraints, $\mathsf{G}\dot{\mathsf{q}}_r = 0$, to ground both ends of the bridge deck. We solve the following dual problem:

$$\mathsf{G}\tilde{\mathsf{M}}_r^{-1}\mathsf{G}^\top\lambda = \mathsf{G}\tilde{\mathsf{M}}_r^{-1}\tilde{\mathsf{f}}_r, \quad \tilde{\mathsf{M}}_r\dot{\mathsf{q}}_r = \tilde{\mathsf{f}}_r - \mathsf{G}^\top\lambda. \tag{4}$$

Let l be the number of rows in the constraint matrix G. We run PCG l times (in parallel) to form the dense LHS matrix $\mathsf{G}\tilde{\mathsf{M}}_r^{-1}\mathsf{G}^\top$, by backsolving with the columns of $\mathsf{G}^\top$, and run PCG once to form the RHS vector. We then solve this linear system for the Lagrange multipliers in $O(l^3)$ time, which is then fed into a final PCG to compute the new velocities. The overall run time is $O(n^\alpha l + l^3)$, where α depends on the scene and is again typically near linear.

4 REDMAX FLEXIBILITY

Having established the efficiency of our method, we now look further at its flexibility. In addition to the combination of implicit and explicit forces acting on reduced and maximal coordinates, we can easily add bilateral and unilateral constraints on both reduced and maximal coordinates. Let G_r, G_m, C_r, and C_m respectively be the reduced bilateral, maximal bilateral, reduced unilateral, and maximal unilateral constraint matrices. These constraints can be used for, *e.g.,* closing loops, attaching FEM nodes to bodies, joint limits, and external contact. (Later in §5, we show how we handle frictional contact constraints.) Applying Gauss's Principle of Least Constraint [Lanczos 2012], these constraints can be incorporated by forming a quadratic program:

$$\begin{aligned} \underset{\dot{\mathsf{q}}_r}{\text{minimize}} \quad & \frac{1}{2}\dot{\mathsf{q}}_r^\top\tilde{\mathsf{M}}_r\dot{\mathsf{q}}_r - \dot{\mathsf{q}}_r^\top\tilde{\mathsf{f}}_r \\ \text{subject to} \quad & \begin{pmatrix}\mathsf{C}_r\\ \mathsf{C}_m\mathsf{J}_{mr}\end{pmatrix}\dot{\mathsf{q}}_r \geq 0, \quad \begin{pmatrix}\mathsf{G}_r\\ \mathsf{G}_m\mathsf{J}_{mr}\end{pmatrix}\dot{\mathsf{q}}_r = 0. \end{aligned} \tag{5}$$

If working at the acceleration level, additional terms involving $\dot{\mathsf{J}}_{mr}$ are required in the constraints.

If only bilateral constraints are present, we can directly form and solve the corresponding KKT linear system [Boyd and Vandenberghe 2004]. For example, when we attach a finite element mesh to the skeleton, we use bilateral constraints that bind certain vertices

to be fixed with respect to the skeleton:

$$\begin{pmatrix} \tilde{\mathsf{M}}_r & 0 & \mathsf{J}_{mr}^\top \mathsf{G}_m^\top \\ 0 & \mathsf{M}_d & \mathsf{G}_d^\top \\ \mathsf{G}_m \mathsf{J}_{mr} & \mathsf{G}_d & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathsf{q}}_r \\ \dot{\mathsf{q}}_d \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{\mathsf{f}}_r \\ \mathsf{f}_d \\ 0 \end{pmatrix}, \tag{6}$$

where the equation of motion of the deformable body is $\mathsf{M}_d \dot{\mathsf{q}}_d = \mathsf{f}_d$, and the constraint for attaching the deformable mesh to the skeleton is $\mathsf{G}_m \mathsf{J}_{mr} \dot{\mathsf{q}}_r + \mathsf{G}_d \dot{\mathsf{q}}_d = 0$. We show an example of this (which also includes hybrid dynamics, described below), in the Starfish scene (Fig. 1c).

When these constraints are applied at the velocity (or acceleration) level, there is an unavoidable constraint drift. We deal with these with the standard Baumgarte [1972] stabilization technique. We emphasize that in practice, stabilizing a few constraint is much easier than stabilizing the whole jointed structure, which is required with maximal coordinate approaches.

Hybrid Dynamics. In *forward* dynamics, we compute the motion given the forces, and in *inverse* dynamics, we compute the forces given the motion. With the RedMax formulation, it is easy to combine these two into *hybrid* dynamics (a term coined by Featherstone), where some DOFs have their accelerations specified, and some DOFs have forces specified. Although the recursive formulation can handle reduced hybrid dynamics [Kim and Pollard 2011], it cannot combine *both reduced and maximal* hybrid dynamics.

Hybrid dynamics can be performed at the acceleration level or at the velocity level, but in this paper, we concentrate on the velocity-level formulation. Let superscript $*$ indicate the subset of joints whose motions are prescribed. Then we can apply a bilateral constraint on the prescribed reduced velocities: ${}^*\mathsf{G}_r \dot{\mathsf{q}}_r = {}^*\dot{\mathsf{q}}_r$, where ${}^*\mathsf{G}_r$ contains the identity matrix in the appropriate blocks so that the prescribed joints will be affected. Similarly, we have ${}^*\mathsf{G}_m \dot{\mathsf{q}}_m = {}^*\dot{\mathsf{q}}_m$ for prescribing maximal velocities, where ${}^*\mathsf{G}_m$ can contain the identity matrix when we want to fully specify the motion of the body, or a point Jacobian (Eq. (S2.4)) when we want to specify the motion of a point on the body. The resulting KKT system is then

$$\begin{pmatrix} \tilde{\mathsf{M}}_r & {}^*\mathsf{G}_r^\top & \mathsf{J}_{mr}^\top {}^*\mathsf{G}_m^\top \\ {}^*\mathsf{G}_r & 0 & 0 \\ {}^*\mathsf{G}_m \mathsf{J}_{mr} & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathsf{q}}_r \\ \lambda_r \\ \lambda_m \end{pmatrix} = \begin{pmatrix} \tilde{\mathsf{f}}_r \\ {}^*\dot{\mathsf{q}}_r \\ {}^*\dot{\mathsf{q}}_m \end{pmatrix}. \tag{7}$$

The required joint torques or maximal wrenches can be computed with the resulting Lagrange multipliers: ${}^*\mathsf{f}_r = {}^*\mathsf{G}_r^\top \lambda_r / h$ and ${}^*\mathsf{f}_m = \mathsf{J}_{mr}^\top {}^*\mathsf{G}_m^\top \lambda_m / h$. With this formulation, we can easily control both maximal and reduced velocities, as shown in our Starfish and Hand examples (Figs. 1c and 2a-2b).

Hyper Reduced Coordinates. We can further reduce the degrees of freedom by chaining more Jacobians. This can be useful, for example, when we want some kinematic coupling between joints. As a concrete example, in a healthy human finger, the two distal joints exhibit *coupled interphalangeal joint motions*—the PIP (first joint away from the knuckle) flexes twice as much as the DIP (the second joint away from the knuckle). (See Figs. 2a-2b.) This is due to the complex arrangement of tendons and ligaments [Leijnse et al. 2010], and can be simulated explicitly at an additional cost [Sueda et al. 2008; Sachdeva et al. 2015]. Alternatively, if we are only interested in the gross kinematics of the finger, we can model this joint-angle relationship directly. To do so, we apply another Jacobian, so that these joint angles are expressed using a single variable. This can be expressed using the following relationship:

$$\begin{aligned} \begin{pmatrix} \dot{\theta}_{\text{PIP}} \\ \dot{\theta}_{\text{DIP}} \end{pmatrix} &= \begin{pmatrix} 2 \\ 1 \end{pmatrix} \dot{\theta} \\ \dot{\mathsf{q}}_r &= \mathsf{J}_{r\text{R}}\, \dot{\mathsf{q}}_{\text{R}}, \end{aligned} \tag{8}$$

where $\dot{\mathsf{q}}_{\text{R}}$ represents the new (hyper) reduced coordinates. If we define

$$\mathsf{J}_{m\text{R}} = \mathsf{J}_{mr}\, \mathsf{J}_{r\text{R}}, \quad \dot{\mathsf{J}}_{m\text{R}} = \dot{\mathsf{J}}_{mr}\, \mathsf{J}_{r\text{R}} + \mathsf{J}_{mr}\, \dot{\mathsf{J}}_{r\text{R}}, \tag{9}$$

then the hyper reduced equation of motion is

$$\mathsf{J}_{m\text{R}}^\top \mathsf{M}_m\, \mathsf{J}_{m\text{R}}\, \ddot{\mathsf{q}}_{\text{R}} = \mathsf{J}_{m\text{R}}^\top \left(\mathsf{f}_m - \mathsf{M}_m \dot{\mathsf{J}}_{m\text{R}} \dot{\mathsf{q}}_{\text{R}} \right). \tag{10}$$

Any combination of bilateral/unilateral and reduced/maximal constraints can be added as before. We show how we can use hyper reduced coordinates in conjunction with reduced and maximal hybrid dynamics with the Hand example (Figs. 2a-2b).

5 FRICTIONAL JOINTS

In this section, we further highlight the flexibility of the RedMax formulation with an efficient algorithm for resolving frictional contact *within* joints. This has many applications including: computing the energy required for robotics; and modeling arthritic joints for biomechanics or animation. In this section, we show how RedMax can be combined with the Staggered Projections (SP) algorithm to take into account the bilateral nature of the joint constraints.

5.1 Review of Staggered Projections

The original Staggered Projections algorithm was developed for solids undergoing unilateral contact constraints with friction [Kaufman et al. 2008]. SP is shown in Alg. 2, slightly modified to match our notation. Since SP was designed for maximal rigid bodies, we remove the m and r (maximal & reduced) subscripts for clarity. There are two quadratic programs (QP) that are solved iteratively: contact and friction. Let $\dot{\mathsf{q}}^{\text{unc}} = \dot{\mathsf{q}}^{\text{prev}} + h\mathsf{M}^{-1}\mathsf{f}$ be the unconstrained velocity, where $\dot{\mathsf{q}}^{\text{prev}}$ is the velocity from the last time step. The contact QP can then be written as:

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2} \alpha^\top \mathsf{N} \mathsf{M}^{-1} \mathsf{N}^\top \alpha - \alpha^\top \mathsf{N} (\dot{\mathsf{q}}^{\text{unc}} + h\mathsf{M}^{-1}\mathsf{f}_\beta) \\ \text{subject to} \quad & \alpha \geq 0, \end{aligned} \tag{11}$$

where α is the contact impulse, N is the contact normal matrix, M is the maximal mass matrix, f is the maximal force, and f_β is the frictional force, which is initially zero. After solving for α, we compute the contact force as $\mathsf{f}_\alpha = -\mathsf{N}^\top \alpha / h$. The frictional QP is:

$$\begin{aligned} \underset{\beta}{\text{minimize}} \quad & \frac{1}{2} \beta^\top \mathsf{T} \mathsf{M}^{-1} \mathsf{T}^\top \beta - \beta^\top \mathsf{T} (\dot{\mathsf{q}}^{\text{unc}} + h\mathsf{M}^{-1}\mathsf{f}_\alpha) \\ \text{subject to} \quad & -\mu\alpha \leq \beta \leq \mu\alpha, \end{aligned} \tag{12}$$

where β is the frictional impulse, T is the contact tangent matrix, and $\mu \geq 0$ is the coefficient of friction. The box constraints can only accommodate a four-sided friction cone—if needed, we can rewrite this constraint to give us a polyhedral cone [Stewart 2000] or a continuous cone [Acary and Brogliato 2008; Li et al. 2018], but we note that for 1 DOF joints, the cone constraint degenerates into a box constraint. After solving for β, we compute the frictional

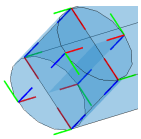
Algorithm 2 Staggered Projections

Fill M ▷ mass matrix
$f_\beta = 0$
while simulating **do**
  Fill f, N, T ▷ force vector, normal and tangent matrices
  $f_\alpha^0 = 0$
  $\dot{q}^{\text{unc}} = \dot{q}^{\text{prev}} + hM^{-1}f$
  while true **do**
    // CONTACT
    Solve contact QP (11) for α
    $f_\alpha = -N^\top \alpha / h$
    // CONVERGENCE CHECK
    if $\|f_\alpha - f_\alpha^0\|_{M^{-1}} \leq \epsilon$ **or** max iterations **then**
      break
    end if
    $f_\alpha^0 = f_\alpha$
    // FRICTION
    Solve friction QP (12) for β
    $f_\beta = -T^\top \beta / h$
  end while
  $\dot{q} = \dot{q}^{\text{prev}} + hM^{-1}(f + f_\alpha + f_\beta)$
end while

force as $f_\beta = -T^\top \beta / h$. These two QPs are solved iteratively until convergence. The convergence rate can be improved by caching the frictional force, f_β, and warm-starting with this cached value at every time step [Kaufman et al. 2008].

5.2 Bilateral Staggered Projections

We extend SP by taking advantage of the bilateral constraints present in articulated rigid body dynamics. The resulting algorithm, which we call Bilateral Staggered Projections (BISP), is much more efficient than SP and can also be combined with SP for handling external frictional contacts, such as between a body and the environment.



BISP has several advantages over SP. First, we do not need collision detection. With BISP, for each joint type (*e.g.,* revolute, spherical, prismatic), we use a small number of implicit contacts at predetermined positions around the joint. For example, for a revolute joint, we assume that the joint geometry is a cylinder, and we populate the two ends of the cylinder with a sparse set of contact points (see inset figure). By changing the parameters of this cylinder, we get different frictional effects. Second, the size of the friction QP decreases significantly, because the friction cone can be represented exactly using box constraints for 1 DOF (revolute and prismatic) joints, which are often the most used joints. Third, the contact QP can be eliminated, since in reduced coordinates, the contact constraints are satisfied automatically. Finally, we obtain faster convergence, since the contacts are more temporally coherent in a bilaterally constrained system.

In the following subsections, we describe how we extend SP to obtain BISP. As stated above, BISP eliminates the need for the contact QP by taking advantage of reduced coordinates, However, we still need the contact Lagrange multipliers, α, when we solve for the

Algorithm 3 Bilateral Staggered Projections

Fill M ▷ mass matrix
$f_\beta = 0$
while simulating **do**
  Fill f, N, T ▷ force vector, normal and tangent matrices
  $f_\alpha^0 = 0$
  while true **do**
    // CONTACT
    Evaluate (15) for f_α ▷ §5.2.1
    while backward traversal **do**
      Distribute f_α to joint ▷ §5.2.2
    end while
    while parallel traversal **do**
      Locally solve (16) for α ▷ §5.2.3
    end while
    // CONVERGENCE CHECK
    if $\|f_\alpha - f_\alpha^0\|_{M^{-1}} \leq \epsilon$ **or** max iterations **then**
      break
    end if
    $f_\alpha^0 = f_\alpha$
    // FRICTION
    Solve friction QP (12) for β
    $f_\beta = -T^\top \beta / h$
  end while
  $\dot{q}_r = \dot{q}_r^{\text{prev}} + h\, M_r^{-1} J_{mr}^\top (\tilde{f}_m + f_\alpha + f_\beta)$
end while

frictional impulses, because α is used as the limits on the friction forces. We compute α in three steps:

- §5.2.1: Compute the joint reaction forces.
- §5.2.2: Distribute this global force into the joints.
- §5.2.3: Compute α locally within each joint.

5.2.1 Compute joint reaction force. We first compute the joint reaction (*i.e.,* constraint) force that would produce the same constrained motion as the one generated using reduced coordinates. We do this by comparing the velocity generated by the reduced solve against the velocity generated by an unconstrained maximal solve. As an illustration, suppose we are running the standard SP algorithm with maximal coordinates. Let $\dot{q}^{\text{unc}} = \dot{q}^{\text{prev}} + hM^{-1}f$, and the corresponding constrained velocity from Eq. (11) be $\dot{q}^{\text{con}}$. We can rearrange the constrained equations of motion to solve for the constraint forces:

$$\begin{aligned} M\dot{q} + N^\top \alpha &= M\dot{q}^{\text{prev}} + h(f + f_\beta) \\ M^{-1}N^\top \alpha &= \underbrace{\dot{q}^{\text{prev}} + hM^{-1}(f + f_\beta)}_{\dot{q}^{\text{unc}}} - \underbrace{\dot{q}}_{\dot{q}^{\text{con}}} \end{aligned} \tag{13}$$

where $\dot{q}^{\text{prev}}$ is the velocity from the last time step. We can rearrange further to obtain the expression for the constraint force, $f_\alpha = -N^\top \alpha / h$:

$$f_\alpha = \frac{1}{h} M \left(\dot{q}^{\text{con}} - \dot{q}^{\text{unc}} \right). \tag{14}$$

What this equation implies is that we can compute the constraint force, f_α, by subtracting the unconstrained velocity from the constrained velocity.

Now we show how BISP uses a similar approach to eliminate the contact QP. As in SP, the current friction force must be taken into account when computing the constrained and unconstrained velocities. In the following equations, since we must now compute both reduced and maximal coordinates, we add back the subscripts m and r. (The contact and friction forces, f_α and f_β, are maximal quantities.) As before, we subtract the unconstrained velocity from the constrained velocity, but now the constrained velocity is computed in reduced coordinates instead of using Eq. (11):

$$\mathrm{f}_\alpha = \frac{1}{h}\mathrm{M}_m\left(\mathrm{J}_{mr}\dot{\mathrm{q}}_r^{\mathrm{con}} - \dot{\mathrm{q}}_m^{\mathrm{unc}}\right) \tag{15a}$$

$$\dot{\mathrm{q}}_m^{\mathrm{unc}} = \mathrm{J}_{mr}\dot{\mathrm{q}}_r^{\mathrm{prev}} + h\mathrm{M}_m^{-1}\left(\mathrm{f}_m + \mathrm{f}_\beta\right) \tag{15b}$$

$$\dot{\mathrm{q}}_r^{\mathrm{con}} = \dot{\mathrm{q}}_r^{\mathrm{prev}} + h\mathrm{M}_r^{-1}\mathrm{J}_{mr}^\top\left(\tilde{\mathrm{f}}_m + \mathrm{f}_\beta\right). \tag{15c}$$

To lighten the notation, we use $\tilde{\mathrm{f}}_m$ in Eq. (15c) to include the quadratic velocity vector from the RHS of Eq. (2).

5.2.2 Distribute contact force to joints. The computed constraint force, f_α, is a global maximal force vector that accounts for all joint reaction forces. Therefore, if we extract a portion of f_α corresponding to a single body, we obtain the *sum* of all the joint reaction forces acting on that body. To compute the Lagrange multipliers for a particular joint, we first need to isolate the joint reaction force from this sum. Fortunately, this can be done in a linear fashion by traversing the joints backward from leaf to root. For a leaf body, there is only one joint force acting on it, and so its portion of f_α is exactly the required joint reaction force. Since this joint reaction force exerts an equal and opposite force on the parent of the leaf, we subtract this force from the parent's portion of f_α and continue the backward traversal.

5.2.3 Compute contact Lagrange multipliers. Once we have the joint reaction forces distributed to each joint, we can compute the contact Lagrange multipliers that generate that joint reaction force. This can be done in parallel, since these are local operations performed for each joint independently of each other. For each joint, we search for a least-squares solution to $\left(\mathrm{N}_i\mathrm{M}_i^{-1}\mathrm{N}_i^\top\right)\alpha_i = h\mathrm{N}_i\mathrm{M}_i^{-1}\mathrm{f}_{\alpha i}$, where the subscript i indicates the blocks corresponding to the i^{th} body. We do not require α_i to be positive, since these "contact" constraints are bilateral—they cannot come apart. To deal with contact indeterminacy [Shin et al. 2016], we add a regularization term, which is critical because otherwise the joint can become arbitrarily tight. For instance, setting $\alpha_i = 1000$ for all contacts will generate the same effective constraint on the joint as setting $\alpha_i = 0.1$. Adding this regularization term, the local linear system becomes:

$$\alpha_i = h\left(\mathrm{N}_i\mathrm{M}_i^{-1}\mathrm{N}_i^\top + \epsilon I\right)^{-1}\left(\mathrm{N}_i\mathrm{M}_i^{-1}\mathrm{f}_{\alpha i}\right). \tag{16}$$

In our experiments, we set ϵ = 1e-6.

After the contact impulses for all the joints, α, are computed, the convergence check and the friction solve are the same as in SP. The step-by-step algorithm is shown in Alg. 3.

5.3 Adding External Constraints

BISP can also take into account external constraints, such as loop-closing (bilateral) constraints or frictional contact (unilateral) constraints with the environment. Alg. 3 is modified as follows to take into account these additional constraints:

• Line 10: To compute the contact force, both the unconstrained and constrained velocities must take into account the additional external constraints. The unconstrained velocity is obtained by solving a maximal system with only the external constraints, ignoring the implicit constraints exerted by the joints. The corresponding constrained velocity is obtained by solving a reduced system with external bilateral constraints, G, and unilateral constraints, C. The difference between these velocities will give us the joint reaction forces. Thus, instead of Eq. (15), we evaluate the following:

$$\mathrm{f}_\alpha = \frac{1}{h}\mathrm{M}_m\left(\mathrm{J}_{mr}\dot{\mathrm{q}}_r^{\mathrm{con}} - \dot{\mathrm{q}}_m^{\mathrm{unc}}\right) \tag{17a}$$

$$\begin{aligned}\min_{\dot{\mathrm{q}}_m}\quad & \frac{1}{2}\dot{\mathrm{q}}_m^\top\mathrm{M}_m\dot{\mathrm{q}}_m - \dot{\mathrm{q}}_m^\top\left(\mathrm{M}_m\mathrm{J}_{mr}\dot{\mathrm{q}}_r^{\mathrm{prev}} + h\left(\mathrm{f}_m + \mathrm{f}_\beta\right)\right)\\ \text{s. t.}\quad & \mathrm{G}_m\dot{\mathrm{q}}_m = 0,\quad \mathrm{C}_m\dot{\mathrm{q}}_m \geq 0\end{aligned} \tag{17b}$$

$$\begin{aligned}\min_{\dot{\mathrm{q}}_r}\quad & \frac{1}{2}\dot{\mathrm{q}}_r^\top\mathrm{M}_r\dot{\mathrm{q}}_r - \dot{\mathrm{q}}_r^\top\left(\mathrm{M}_r\dot{\mathrm{q}}_r^{\mathrm{prev}} + h\mathrm{J}_{mr}^\top\left(\tilde{\mathrm{f}}_m + \mathrm{f}_\beta\right)\right)\\ \text{s. t.}\quad & \mathrm{G}_m\mathrm{J}_{mr}\dot{\mathrm{q}}_r = 0,\quad \mathrm{C}_m\mathrm{J}_{mr}\dot{\mathrm{q}}_r \geq 0,\end{aligned} \tag{17c}$$

where $\dot{\mathrm{q}}_m^{\mathrm{unc}}$ and $\dot{\mathrm{q}}_r^{\mathrm{con}}$ are the solutions of the two QPs ((17b) & (17c)). The loop-closing constraint reaction forces are computed as $\mathrm{f}_\lambda = -\mathrm{J}_{mr}^\top\mathrm{G}_m^\top\lambda/h$, where λ is the vector of Lagrange multipliers corresponding to the loop-closing constraints, $\mathrm{G}_m\mathrm{J}_{mr}\dot{\mathrm{q}}_r = 0$, from the minimization for $\dot{\mathrm{q}}_r^{\mathrm{con}}$. The *maximal* QP in Eq. (17b) may seem expensive to solve (all other QPs are in *reduced* coordinates), but usually, the number of external constraints is much smaller than the number of joint constraints. Therefore, we can solve this maximal QP in its dual form instead, which is much smaller. For example, for the Klann mechanism with 6 legs (Fig. 1d), the dual QP is of size at most 30 with only box constraints.

• Line 15: We need to compute the contact forces due to the loop-closing joint constraints, by again solving a small linear system (16). These small linear systems are solved for each joint *and* for each loop-closing joint constraint.

• Line 26: To compute the final velocity, we solve a quadratic program that takes into account the external constraints:

$$\begin{aligned}\min_{\dot{\mathrm{q}}_r}\quad & \frac{1}{2}\dot{\mathrm{q}}_r^\top\mathrm{M}_r\dot{\mathrm{q}}_r - \dot{\mathrm{q}}_r^\top\left(\mathrm{M}_r\dot{\mathrm{q}}_r^{\mathrm{prev}} + h\mathrm{J}_{mr}^\top\left(\tilde{\mathrm{f}}_m + \mathrm{f}_\alpha + \mathrm{f}_\beta\right)\right)\\ \text{s. t.}\quad & \mathrm{G}_m\mathrm{J}_{mr}\dot{\mathrm{q}}_r = 0,\quad \mathrm{C}_m\mathrm{J}_{mr}\dot{\mathrm{q}}_r \geq 0.\end{aligned} \tag{18}$$

With these three changes, any combination of external bilateral and unilateral constraints can be incorporated into BISP.

6 RESULTS

We implemented our system in C++ and ran the simulations on a consumer desktop with an Intel Core i7-7700 CPU @ 3.6 Ghz and 16 GB of RAM. We use Eigen for dense linear algebra, Pardiso for sparse linear solves, and Mosek for quadratic programs. For the direct solver, the system indices are ordered backward from leaf to

minimize fill-ins [Negrut et al. 1997]. For PCG, the stopping tolerance is set to relative residual of 1e-6. Please see the supplemental video for the animations.

• Starfish (Fig. 1c): We model a starfish with a skeleton consisting of 20 joints and a coarse FEM mesh consisting of 221 vertices. For display and collision, we embed a fine mesh with 7909 vertices inside the coarse simulation mesh. We use co-rotated elasticity, but any material model can be used [Sifakis and Barbic 2012]. We use RedMax hybrid dynamics to animate the starfish—we procedurally prescribe some of the joints as well as some specific points on the skeleton. The rest of the skeleton and the FEM mesh are passively simulated with fully implicit two-way coupling.

• Hand (Fig. 2): The two distal joints of a healthy human finger exhibits coupled interphalangeal joint motions [Leijnse et al. 2010]. Specifically, the most distal joint (DIP) usually flexes by half the joint angle of the second most distal joint (PIP). We model a human hand where we use hyper reduced coordinates on the DIP/PIP of the four fingers. We then animate the hand with RedMax hybrid dynamics. The fingertip positions are prescribed using maximal inverse dynamics, and at the same time, the elbow angle is prescribed using reduced inverse dynamics.

• Hagfish (Fig. 3): The hagfish is unique in that it forms a knot with its body to provide leverage to its head as it feeds. Using the parameters taken from the literature [Evans et al. 2018], we model a realistic hagfish that can be used as a testbed for studying how a hagfish controls its body to form a knot, which is currently an open scientific problem. We model the animal by placing Z-revolute and XY-universal joints in an alternating fashion, for a total of 100 joints. We set the length, mass, radii, and joint limits with data taken from real measurements. We perform self-collision detection by passing a spline curve through the length of the animal and finding the colliding points with Newton's method. The collisions are turned into inequality constraints, resulting in a quadratic program (Eq. (5)). We do not model friction, since hagfish are slimy.

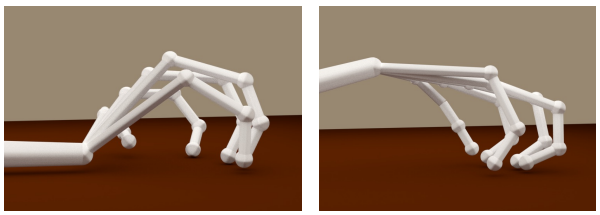


Fig. 2. Hand simulation with coupled interphalangeal joint motions.

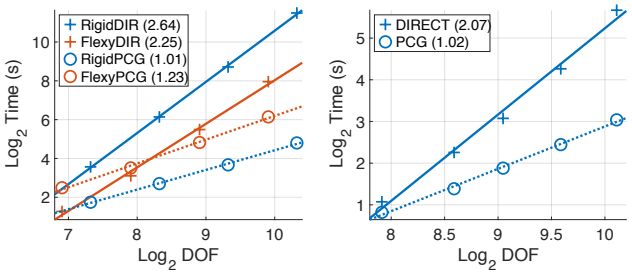
(a) (b) (c) (d)

Fig. 3. (a-b) A knotting Hagfish, with its head in a jar. (c-d) Our simulation.

• Bridge (Figs. 1a & 4a): To show the scalability of our preconditioner, we model a cable-stayed bridge with a fan design. The towers and the deck are composed of a sequence of bodies connected by revolute joints, with cables attaching the deck to the towers. We vary the number of deck and tower pieces, as listed in Table 2. We solve this system with PCG using our preconditioner and with an off-the-shelf direct solver. When the towers are modeled as rigid, PCG converges in *a single iteration*. In this case, our block diagonal preconditioner becomes exactly the inverse of the system matrix, since the tower bodies are removed from the system, and so only the block diagonal portions of the local stiffness matrices enter the system matrix. When we make the tower joints flexible and apply an exaggerated weight (see accompanying video), our preconditioner still shows good scaling, though the direct method is faster initially. Table 2 & Fig. 4a show the timing result of the simulation. The slope

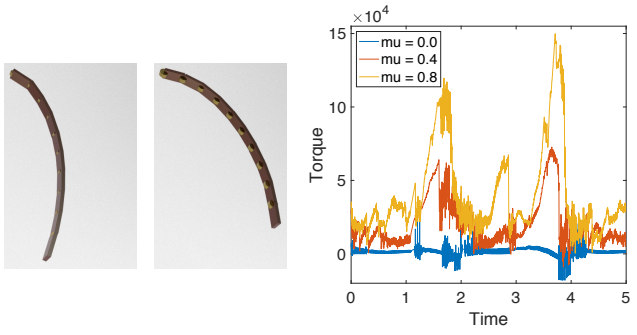
Table 2. Performance table for Bridge and Umbrella. *DOF*: Degrees of freedom. *Direct & PCG*: Wall-clock times in seconds to simulate five units of scene time. *Iters*: Average number of PCG iterations per step. For Bridge, there are 2 sub scenes: [Rigid] is with rigid towers; [Flexy] is with flexible towers and an exaggerated weight.

	DOF	*Direct*	*PCG*	*Iters*
Bridge [Rigid]	80	2.8	2.0	4.0
	160	11.8	3.4	4.0
	320	70.5	6.5	4.0
	640	420.3	12.7	4.0
	1280	2884.7	27.8	4.0
Bridge [Flexy]	60	1.0	2.7	104.7
	120	2.4	5.6	136.6
	240	8.6	11.4	153.5
	480	44.9	28.3	196.3
	960	248.7	70.6	241.9
Umbrella	241	2.1	1.8	37.0
	385	4.8	2.6	36.0
	529	8.4	3.4	36.7
	769	19.1	5.1	38.5
	1105	50.1	8.4	39.5



(a) Bridge (b) Umbrella

Fig. 4. $\log_2 - \log_2$ plot of Time vs. DOFs for Bridge and Umbrella. Solid lines are for a direct solver, and dashed lines are for our preconditioned solver. The computed slopes indicate that with our preconditioner, the runtime performance is near linear.



(a) Chain (R) (b) Chain ($2R$) (c) Klann torque

Fig. 5. (a) Chain composed of revolute joints. (b) With a larger radius, the chain stops earlier. (c) Klann: graph of torque vs. time, with various joint friction coefficients.

of the fitted lines of the $\log_2 - \log_2$ plot of time vs. DOFs shows the empirical order of each approach. When the tower stiffness is infinite, PCG takes $O(n^{1.01})$ time, and when the towers are flexible, PCG takes $O(n^{1.23})$ time. As a comparison, a direct solver takes $O(n^{2.64})$ and $O(n^{2.25})$ time, respectively.

• Umbrella (Figs. 1b & 4b): As another scaling test, we model a deployable umbrella. The root body is the *tube*, with 8 *ribs* attached to its tip. The 8 ribs are pushed open by the 8 *stretchers*. The stretchers are attached to the *runner*, which has a prismatic joint with respect to the tube. Both the ribs and the stretchers are modeled using a sequence of universal joints, and are connected to each other by bilateral constraints. Springs are placed between the 8 ribs to model the canopy. We vary the number of bodies in the ribs and the stretchers. Table 2 & Fig. 4b show the number of DOFs, the average iteration count for PCG, and the wall-clock time for PCG and the direct solver to simulate a 1 second scene. The empirical orders of the two methods are $O(n^{1.01})$ and $O(n^{2.07})$ for PCG and direct, respectively. Our unoptimized PCG is initially slower than the direct solver, but when n is large, the PCG becomes faster.

• Chain (Figs. 5a & 5b): This scene shows the frictional effect due to changing the geometry of the joint. We initialize the scene so that the chain starts horizontally and falls under gravity, with the axis of rotation oriented 45° from the direction of gravity. Since more weight must be supported by bodies closer to the root, the contact force, and thus the force of friction, is stronger at the root compared to the tip. This makes the chain stop rotating starting from the root rather than at the tip. When we increase the joint radius, even with the same coefficient of friction, the force of friction increases since the joint is able to apply more torque. When Staggered Projections is used, the simulation takes an order of magnitude longer to complete because: (1) with maximal coordinates, position stabilization is required, and this can have an adverse effect on the iteration count; and (2) SP requires two global QPs, whereas BISP requires only one.

• Klann (Figs. 1d & 5c): Our friction solver can handle loop-closure and contact constraints. We build a walking machine with a Klann linkage for each of the 6 limbs. Each limb has 5 revolute joints and 2 loop-closure constraints. Since each loop-closure constraint removes 2 DOFs, each limb has $5 - 2 \cdot 2 = 1$ effective DOF. In total, there are 26 internal joint DOFs with 24 bilateral constraints and (up to) 6 unilateral constraints. The 2 remaining DOFs are driven with inverse dynamics to have a constant rotational speed. We use $\mu = 0.8$ for floor friction. We run several simulations, increasing μ for loop closure and joint constraints from 0.0 to 0.8. Fig. 5c shows the amount of torque required to drive the mechanism as we increase this frictional coefficient. As we expect, more torque is required when there is more friction within the joints. Interestingly, when the $\mu = 0$, the motor does some negative work—the limbs try to push the motor forward, but the motor pushes back.

7 CONCLUSION

We introduced an efficient and flexible approach for computing the dynamics of articulated rigid bodies. Unlike prior approaches that require $O(n^3)$ time, our approach maintains near linear performance, even in the presence of maximal stiffness matrix used by a linearly implicit integrator. In some simulation scenarios, our preconditioned solver converges in a single iteration. Our approach also provides flexibility, allowing us to mix and match implicit and explicit forces in both reduced and maximal coordinates, as well as bilateral and unilateral constraints in either coordinates. We showed this flexibility with several results including those that use hybrid dynamics in both coordinates and fully two-way coupled dynamics of articulated and deformable bodies. Finally, we showed how our approach can be efficiently integrated into a friction solver that can incorporate friction inside the joints with loop closure as well as external contact constraints. We include with this work reference implementations of RedMax written in C++ and object-oriented MATLAB as supplemental material.[2]

7.1 Limitations & Future Work

Although the theoretical runtime of factorization methods are $O(n^3)$ [De Jalon and Bayo 2012], in practice, they exhibit better asymptotic behavior depending on the sparsity pattern of the system matrix. When the scene has many branches, the system often becomes very sparse, and these methods become subquadratic, with very small overhead compared to our PCG method. Automatically detecting when to switch between the two methods would be of practical interest. This is especially true since we have only shown *empirically* that PCG takes a sublinear number of iterations using our preconditioner, without a formal proof.

We have not taken into account parallelization or GPU implementations. Although some parts of our approach could be easily parallelizable (*e.g.,* each branch in the tree topology can be processed in parallel), we have not investigated how best to optimize our approach. Existing $O(n)$ and $O(n^3)$ methods have shown good parallelizability (*e.g.,* [Avello et al. 1993; Negrut et al. 1997]), and so we believe it is worthwhile to explore similar techniques that work even with the inclusion of the stiffness matrix.

Our efficient preconditioner is applicable only when the number of deformable DOFs is relatively small compared to the rigid DOFs. We hope to explore the special case of quasistatic deformable objects, where the deformable DOFs are eliminated from the system. In these cases, we may be able to utilize our efficient method for

[2] https://github.com/sueda/redmax

the coupled simulation of rigid and deformable objects. It would also be interesting to apply the strategy proposed by Redon et al. [2005] on automatically figuring out which joints should be activiated/deactivated at runtime.

Our Bilateral Staggered Projections algorithm is an extension of the Staggered Projections algorithm [Kaufman et al. 2008], which iteratively solves a pair of coupled quadratic programs to resolve the frictional force. It would be interesting to also extend the popular approximate frictional contact model by Anitescu and Hart [2004], which uses only a single QP.

Finally, extending RedMax to other simulation techniques such as Projective and Position-Based Dynamics [Bouaziz et al. 2014; Müller et al. 2007] would be useful for real-time applications.

ACKNOWLEDGMENTS

We thank Feras Khemakhem for rendering the results and the anonymous reviewers for their comments. This work was sponsored in part by the National Science Foundation (CAREER-1846368).

REFERENCES

V. Acary and B. Brogliato. 2008. *Numerical Methods for Nonsmooth Dynamical Systems: Applications in Mechanics and Electronics.* Springer Science & Business Media.

M. Anitescu and G. D. Hart. 2004. A Fixed-point Iteration Approach for Multibody Dynamics with Contact and Small Friction. *MATH. PROG.* 101, 1 (2004), 3–32.

A. Avello, J. M. Jiménez, E. Bayo, and J. G. de Jalón. 1993. A Simple and Highly Parallelizable Method for Real-time Dynamic Simulation Based on Velocity Transformations. *Comput. Methods in Appl. Mech. Eng.* 107, 3 (1993), 313–339.

D.-S. Bae and E. J. Haug. 1987a. A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems. *J STRUCT MECH* 15, 3 (1987), 359–382.

D.-S. Bae and E. J. Haug. 1987b. A Recursive Formulation for Constrained Mechanical System Dynamics: Part II. Closed Loop Systems. *J STRUCT MECH* 15, 4 (1987), 481–506.

D. Baraff. 1996. Linear-time Dynamics Using Lagrange Multipliers. In *Annual Conference Series (Proc. SIGGRAPH).* 137–146.

D. Baraff and A. Witkin. 1998. Large Steps in Cloth Simulation. In *Annual Conference Series (Proc. SIGGRAPH).* 43–54.

J. Baumgarte. 1972. Stabilization of Constraints and Integrals of Motion in Dynamical Systems. *Comput. Methods in Appl. Mech. Eng.* 1 (Jun 1972), 1–16.

F. Bertails. 2009. Linear Time Super-Helices. *Computer Graphics Forum (Proc. Eurographics)* 28, 2 (May 2009), 417–426.

S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014).

S. Boyd and L. Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press.

M. B. Cline and D. K. Pai. 2003. Post-stabilization for Rigid Body Simulation with Contact and Constraints. In *IEEE Int. Conf. Robot. Autom.*, Vol. 3. 3744–3751.

J. G. De Jalon and E. Bayo. 2012. *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge.* Springer Science & Business Media.

S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. 2007. OpenSim: Open-source Software to Create and Analyze Dynamic Simulations of Movement. *IEEE T BIO-MED ENG* 54, 11 (2007), 1940–1950.

E. D. Demaine and J. O'Rourke. 2008. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra* (reprint ed.). Cambridge University Press.

M. Deuss, D. Panozzo, E. Whiting, Y. Liu, P. Block, O. Sorkine-Hornung, and M. Pauly. 2014. Assembling Self-supporting Structures. *ACM Trans. Graph.* 33, 6, Article 214 (Nov. 2014).

E. Drumwright. 2012. Fast Dynamic Simulation of Highly Articulated Robots with Contact Via $\theta(n^2)$ Time Dense Generalized Inertia Matrix Inversion. In *Int. Conf. on Sim., Model., & Prog. for Auton. Robots.* Springer, 65–76.

E. Drumwright and D. A. Shell. 2010. Modeling Contact Friction and Joint Friction in Dynamic Robotic Simulation Using the Principle of Maximum Dissipation. In *Algorithmic Foundations of Robotics IX.* Springer, 249–266.

E. Evans, Y. Hwang, S. Sueda, and T. A. Uyeno. 2018. Estimating Whole Body Flexibility in Pacific Hagfish. In *The Society for Integrative & Comparative Biology.*

R. Featherstone. 1983. The Calculation of Robot Dynamics Using Articulated-body Inertias. *INT J ROBOT RES* 2, 1 (1983), 13–30.

S. Hadap. 2006. Oriented Strands: Dynamics of Stiff Multi-body System. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim. (SCA '06).* 91–100.

F. Hernandez, C. Garre, R. Casillas, and M. A. Otaduy. 2011. Linear-Time Dynamics of Characters with Stiff Joints. In *V Ibero-American Symposium on Computer Graphics (SIACG 2011).* The Eurographics Association and Blackwell Publishing Ltd.

S. Jain and C. K. Liu. 2011. Controlling Physics-based Characters Using Soft Contacts. *ACM Trans. Graph.* 30, 6, Article 163 (Dec. 2011).

D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai. 2008. Staggered Projections for Frictional Contact in Multibody Systems. *ACM Trans. Graph.* 27, 5, Article 164 (Dec 2008).

J. Kim. 2012. *Lie Group Formulation of Articulated Rigid Body Dynamics.* Technical Report. Carnegie Mellon University.

J. Kim and N. S. Pollard. 2011. Fast Simulation of Skeleton-driven Deformable Body Characters. *ACM Trans. Graph.* 30, 5, Article 121 (Oct. 2011).

C. Lanczos. 2012. *The Variational Principles of Mechanics* (4 ed.). Dover Publications.

J. Leijnse, P. Quesada, and C. Spoor. 2010. Kinematic Evaluation of the Finger's Interphalangeal Joints Coupling Mechanism—variability, Flexion–extension Differences, Triggers, Locking Swanneck Deformities, Anthropometric Correlations. *Journal of Biomechanics* 43, 12 (2010), 2381–2393.

J. Li, G. Daviet, R. Narain, F. Bertails-Descoubes, M. Overby, G. E. Brown, and L. Boissieux. 2018. An Implicit Frictional Contact Solver for Adaptive Cloth Simulation. *ACM Trans. Graph.* 37, 4, Article 52 (July 2018).

L. Liu, K. Yin, B. Wang, and B. Guo. 2013. Simulation and Control of Skeleton-driven Soft Body Characters. *ACM Trans. Graph.* 32, 6, Article 215 (Nov. 2013).

M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. 2007. Position Based Dynamics. *J VIS COMMUN IMAGE R* 18, 2 (2007), 109–118.

D. Negrut, R. Serban, and F. A. Potra. 1997. A Topology-based Approach to Exploiting Sparsity in Multibody Dynamics: Joint Formulation. *J STRUCT MECH* 25, 2 (1997), 221–241.

E. P. Popov, A. F. Vereshchagin, and S. L. Zenkevich. 1978. Robot Manipulators: Dynamics and Algorithms.

E. Quigley, Y. Yu, J. Huang, W. Lin, and R. Fedkiw. 2018. Real-Time Interactive Tree Animation. *IEEE TVCG* 24, 5 (May 2018), 1717–1727.

S. Redon, N. Galoppo, and M. C. Lin. 2005. Adaptive Dynamics of Articulated Bodies. *ACM Trans. Graph.* 24, 3 (July 2005), 936–945.

R. E. Roberson. 1966. A Dynamical Formalism for an Arbitrary Number of Interconnected Rigid Bodies with Reference to the Problem of Satellite Attitude Control. *Proc. 3rd Congr. of Int. Fed. Automatic Control* 1 (1966), 46D1–46D8.

P. Sachdeva, S. Sueda, S. Bradley, M. Fain, and D. K. Pai. 2015. Biomechanical Simulation and Control of Hands and Tendinous Systems. *ACM Trans. Graph.* 34, 4, Article 42 (July 2015).

L. Sciavicco and B. Siciliano. 2012. *Modelling and Control of Robot Manipulators.* Springer Science & Business Media.

R. Serban, D. Negrut, E. J. Haug, and F. A. Potra. 1997. A Topology-based Approach for Exploiting Sparsity in Multibody Dynamics in Cartesian Formulation. *J STRUCT MECH* 25, 3 (1997), 379–396.

A. A. Shabana. 2013. *Dynamics of Multibody Systems.* Cambridge University press.

H. V. Shin, C. F. Porst, E. Vouga, J. Ochsendorf, and F. Durand. 2016. Reconciling Elastic and Equilibrium Methods for Static Analysis. *ACM Trans. Graph.* 35, 2, Article 13 (Feb. 2016).

T. Shinar, C. Schroeder, and R. Fedkiw. 2008. Two-way Coupling of Rigid and Deformable Bodies. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 95–103.

E. Sifakis and J. Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses.*

D. E. Stewart. 2000. Rigid-Body Dynamics with Friction and Impact. *SIAM Rev.* 42, 1 (March 2000), 3–39.

S. Sueda, A. Kaufman, and D. K. Pai. 2008. Musculotendon Simulation for Hand Animation. *ACM Trans. Graph.* 27, 3, Article 83 (Aug. 2008).

M. Tournier, M. Nesme, B. Gilles, and F. Faure. 2015. Stable Constrained Dynamics. *ACM Trans. Graph.* 34, 4, Article 132 (July 2015).

M. W. Walker and D. E. Orin. 1982. Efficient Dynamic Computer Simulation of Robotic Mechanisms. *J DYN SYST-T ASME* 104, 3 (1982), 205–211.

J. M. Wang, S. R. Hamner, S. L. Delp, and V. Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-based Actuators and Objectives. *ACM Trans. Graph.* 31, 4, Article 25 (July 2012).

T. M. Wasfy and A. K. Noor. 2003. Computational Strategies for Flexible Multibody Systems. *APPL MECH REV* 56, 6 (2003), 553–613.

E. Whiting, H. Shin, R. Wang, J. Ochsendorf, and F. Durand. 2012. Structural Optimization of 3D Masonry Buildings. *ACM Trans. Graph.* 31, 6, Article 159 (Nov. 2012).

Y. Zhou, S. Sueda, W. Matusik, and A. Shamir. 2014. Boxelization: Folding 3D Objects into Boxes. *ACM Trans. Graph.* 33, 4, Article 71 (July 2014).