ConJac: Large Steps in Dynamic Simulation

Nicholas J. Weidner
Texas A&M University

Theodore Kim
Yale University

Shinjiro Sueda
Texas A&M University

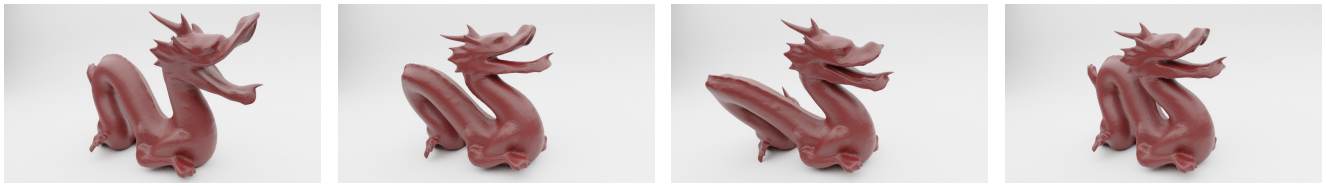


Figure 1: ConJac Dragon. Pulling portions of the mesh generates lively motion using only a small number of dynamic nodes.

ABSTRACT

We present a new approach that allows large time steps in dynamic simulations. Our approach, ConJac, is based on condensation, a technique for eliminating many degrees of freedom (DOFs) by expressing them in terms of the remaining degrees of freedom. In this work, we choose a subset of nodes to be *dynamic* nodes, and apply condensation at the velocity level by defining a linear mapping from the velocities of these chosen dynamic DOFs to the velocities of the remaining *quasistatic* DOFs. We then use this mapping to derive reduced equations of motion involving only the dynamic DOFs. We also derive a novel stabilization term that enables us to use complex nonlinear material models. ConJac remains stable at large time steps, exhibits highly dynamic motion, and displays minimal numerical damping. In marked contrast to subspace approaches, ConJac gives exactly the same configuration as the full space approach once the static state is reached. ConJac works with a wide range of moderate to stiff materials, supports anisotropy and heterogeneity, handles topology changes, and can be combined with existing solvers including rigid body dynamics.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; *Simulation by animation.*

KEYWORDS

Physical simulation, deformation, finite elements

ACM Reference Format:
Nicholas J. Weidner, Theodore Kim, and Shinjiro Sueda. 2020. ConJac: Large Steps in Dynamic Simulation. In *Motion, Interaction and Games (MIG '20), October 16–18, 2020, Virtual Event, SC, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3424636.3426901

MIG '20, October 16–18, 2020, Virtual Event, SC, USA

ACM ISBN 978-1-4503-8171-0/20/10…$15.00
https://doi.org/10.1145/3424636.3426901

1 INTRODUCTION

Physics-based simulation of dynamic deformable objects has a long history in computer graphics. Starting with the work by Terzopoulos et al. [1987], algorithms for physics-based animation have steadily become an integral part of the visual effects pipeline. Over the years, various improvements have been made, including: novel energy formulations [Smith et al. 2018], inversion recovery/safety [Irving et al. 2004; Kim et al. 2019], novel Eulerian/Largrangian formulations [Jiang et al. 2016; Levin et al. 2011], and completely new time stepping schemes [Bouaziz et al. 2014; Müller et al. 2007].

Computational efficiency is one of the most important aspects of simulation. Real-time applications such as games and virtual surgery have strict computational budgets for physics, while offline applications such as movies need efficiency so that artists can quickly iterate on designs. However, efficiency comes at a price. Various works have made dynamic simulation of deformable objects extremely efficient, but they inescapably introduce limitations. To tackle this issue, we introduce a novel, reduced coordinate approach that has the following desirable properties:

- Reproduces *exactly the same static configuration* as the standard finite element (FE) approach.
- Supports complex nonlinear materials, including heterogeneity, anisotropy, and biomechanical soft tissues.
- Does not require any precomputation.
- Supports topology changes.
- Retains dynamic motion at large time steps, without suffering from excessive numerical damping.
- Can be combined with existing frameworks, including rigid body dynamics, into a fully two-way coupled simulation.

Existing works fail with respect to at least one of these properties. The virtual surgery simulator of Bro-Nielsen and Cotin [1996] is highly efficient and produces the same static configuration as the full FE method, which is useful for predicting the behavior of a virtual organ. However, it only supports relatively small deformations, because only linear materials can be factorized as a precomputation. In one of the seminal works on cloth simulation, Baraff and Witkin [1998] greatly increased the efficiency of dynamics simulations by introducing a linearly implicit integration method that allowed large time steps. However, this approach fails to retain dynamics under large time steps due to numerical damping. One of the most important approaches to improving efficiency is subspace dynamics [An et al. 2008; Barbič and James 2005; Choi and Ko 2005; Pan et al.

2015; Pentland and Williams 1989; Teng et al. 2015]. These methods achieve massive speed ups, but sacrifice local detail because the subspace dimension must be kept at a minimum. They also require precomputation, and cannot reproduce the same solution as FE unless a prohibitively large subspace is used.

Our approach is based on condensation [Paz 1989], a technique for eliminating many degrees of freedom (DOFs) by expressing them in terms of the remaining DOFs. With ConJac, short for *Con*densation *Jac*obian, we apply condensation at the velocity level—a significant departure from previous work [Bro-Nielsen and Cotin 1996; Gao et al. 2014; Paz 1989; Teng et al. 2015; Wilson 1974]. We select a "dynamic" subset of nodes as the true DOFs of the system, and the remaining "quasistatic" nodes are assumed to follow the dynamic nodes in a quasistatic fashion.[1] More specifically, ConJac expresses the velocities of the quasistatic nodes as a linear function of the dynamic nodes by leveraging the condition that the net force acting on each quasistatic node vanishes. We also derive a novel stabilization term that allows ConJac to be used with an arbitrary material model. Previous work was limited to linear materials.

We show that most of the important dynamics of an object are captured by simulating just a few key dynamic nodes, and the remainder can be handled quasistatically. We simulate a bar stretching, compressing, bending, and twisting with only a few (1-4) dynamic nodes placed along the central axis. We are also able to simulate the dynamics of a dragon being pulled in various locations, and a bunny being dropped on the floor, each with only 8 dynamic nodes. The ConJac approach remains stable with large time steps because the quasistatic nodes cannot move independently, which effectively removes the small vibrations that can destabilize standard FE simulators. With ConJac, a strong force suddenly applied to a node is instantaneously propagated to the dynamic nodes, eliminating the numerical wave that would force a full FE simulator to take small time steps.

ConJac is a method for reducing the DOFs of a system, and so it is not tied to a specific time integrator. In this paper, we showcase the strengths of ConJac using the popular linearly implicit integration scheme [Baraff and Witkin 1998]. We show that with a linearly implicit scheme, ConJac is computationally inexpensive, requiring only one linear solve per time step, but does not suffer excessively from numerical damping and retains all of the advantages listed earlier in the introduction.

2 RELATED WORK

Simulation of deformable objects is a well-studied subject in computer animation, and we refer the reader to excellent existing surveys and tutorials [Nealen et al. 2006; Sifakis and Barbic 2012].

Our method is based on condensation, a technique from structural engineering [Guyan 1965; Irons 1965; Wilson 1974]. Originally developed for static vibrational analysis, condensation has been extended to include dynamics [Paz 1989]. With these classical condensation approaches, a global generalized eigenvalue problem is solved for the reduced modes of the structure. In our work, we use condensation to derive a linear mapping of the velocities rather than to compute the modes.

[1] Previous works have called these "external/internal" or "master/slave" nodes.

Several previous works in computer graphics are motivated by condensation. These methods use the stiffness matrix to couple specially chosen dynamic DOFs to the remaining quasistatic nodes. Our work is closely related to the work by Gao et al. [2014] on Steklov-Poincaré skinning. They achieve impressive volumetric effects for skinning using only the surface degrees of freedom, but is limited to quasistatics and corotational elasticity. The same authors later developed a "macroblock" solver for grid-based discretizations, also using a stiffness matrix reduction [Mitchell et al. 2016]. By solving the macroblocks in parallel and efficiently aggregating, they quickly compute a deformation that matches the output of a standard FE solver. However, they again rely on linear (corotational) material that can be precomputed. Furthermore, stiff springs are used to couple deformable objects to rigid bodies, which may reduce the time step or introduce unwanted numerical damping.

One of the most important and popular approaches to improving efficiency is subspace dynamics [An et al. 2008; Barbič and James 2005; Choi and Ko 2005; Li et al. 2014; Pan et al. 2015; Pentland and Williams 1989]. Rather than simulating the full space of vertex DOFs, dynamics are performed over a reduced set of DOFs. To address artifacts that arise from the global support of subspace basis functions, researchers have explored domain decompositions where subspaces are computed per domain. To stitch these domains together, Barbič and Zhao [2011] used locally aligned rigid frames, while Kim and James [2011] used penalty forces. These methods can achieve massive speed ups, but sacrifice local detail because the subspace dimension must be kept at a minimum. They also require precomputations such as modal analysis and cubature optimization, so changing object topologies are challenging. Finally, they generally do not reproduce the full FE solution unless the subspace is prohibitively large.

Condensation has also been combined with subspace dynamics. Traditionally, only linear materials could be used, but Teng et al. [2015] efficiently performed subspace condensation at runtime, allowing nonlinear materials to also be used. However, the overall limitations remain. The subspace must be carefully constructed, and while the condensation allows objectionable artifacts to be avoided, the final deformation does not match the full FE solution.

Recently, Xian et al. [2019] introduced a multigrid-based method to solve for deformation dynamics in the full space, and achieved over 40 FPS on a mesh with over 60k vertices. However, they inherit common limitations of multigrid methods. Without significant extensions, it is not possible to support topological changes, complex materials (heterogeneity and anisotropy), and two-way coupling with rigid body dynamics.

Finally, a number of efficient time stepping schemes have been introduced by graphics researchers. Recently, Li et al. [2019] introduced a domain-decomposed optimization method for implicit numerical time integration. In the past two decades, Position-Based Dynamics [Müller et al. 2007], Projective Dynamics [Bouaziz et al. 2014], and ADMM [Narain et al. 2016; Zhang et al. 2019] have become popular, efficient alternatives to the standard time stepping schemes. Although initially quite limited in terms of available materials and constraints, these methods have become quite general and flexible. These time stepping schemes work well, but are monolithic, and would require a complete rewrite of existing formulations to make them work together. Our work is instead based on a simple

mapping of velocities, which can be incorporated into a wide range of existing explicit and implicit integrators.

3 CONJAC DYNAMICS

We begin with a high-level, didactic description of ConJac in action. Imagine a vertical string discretized as a sequence of 1D nodes (*i.e.,* they can only move vertically). We fix the top node and pick the bottom node to be the *dynamic* node. The remaining nodes in the middle are labeled as *quasistatic* nodes. If we know the material properties of the string (*e.g.,* zero rest-length springs), then by assuming that the net force on each quasistatic node remains zero, we can calculate the position and velocities of all these quasistatic nodes from the position and velocity of the single dynamic node at the bottom of the string.

In this section, we will formalize this approach by deriving the linear mapping between the quasistatic and dynamic nodes of a volumetric solid composed of an *arbitrary nonlinear material*. We will then derive equations of motion that allow us to simulate the object using only the dynamic DOFs. The remaining nodes are simulated quasistatically, so the final resting configuration exactly matches the result of a full, non-reduced FE simulator.

3.1 ConJac Mapping

Once again, we select a set of *dynamic* nodes that are the exposed degrees of freedom of the system. The remaining *quasistatic* nodes move so that their net force always resolves to zero. The ConJac framework uses the linear mapping that enforces this condition between the dynamic and quasistatic nodal velocities:

$$\mathrm{v}_q = \mathrm{J}_{qd}\mathrm{v}_d, \tag{1}$$

where J_{qd} is the Jacobian term that we will derive in the rest of this section. Given any velocities of the dynamic nodes, v_d, this mapping allows us to compute the velocities of the quasistatic nodes, v_q.

The derivation of J_{qd} in Eq. 1 starts with a linearization of the forces, popularized by Baraff and Witkin [1998] and extensively used by other researchers [Nealen et al. 2006]. We approximate the implicit force at the next time step as:

$$\mathrm{f} = \mathrm{f}^0 + \mathrm{K}^0(\mathrm{x} - \mathrm{x}^0), \tag{2}$$

where the superscript 0 denotes the quantities at the current time step, and $\mathrm{K} = \partial \mathrm{f}/\partial \mathrm{x}$ is the tangent stiffness matrix. Substituting the next velocity as $\mathrm{v} = (\mathrm{x} - \mathrm{x}^0)/h$, where h is the step size, we obtain:

$$\mathrm{f} = \mathrm{f}^0 + \mathrm{K}^0 h\mathrm{v}. \tag{3}$$

We follow previous condensation work [Bro-Nielsen and Cotin 1996; Gao et al. 2014; Paz 1989; Teng et al. 2015; Wilson 1974] and partition each of the terms into dynamic and quasistatic quantities:

$$\begin{pmatrix}\mathrm{f}_d\\ \mathrm{f}_q\end{pmatrix} = \begin{pmatrix}\mathrm{f}^0_d\\ \mathrm{f}^0_q\end{pmatrix} + h\begin{pmatrix}\mathrm{K}^0_{dd} & \mathrm{K}^0_{dq}\\ \mathrm{K}^0_{qd} & \mathrm{K}^0_{qq}\end{pmatrix}\begin{pmatrix}\mathrm{v}_d\\ \mathrm{v}_q\end{pmatrix}. \tag{4}$$

Since we are interested in applying the zero net-force condition on the quasistatic nodes, we extract the bottom row of Eq. 4. After moving f^0_q and h to the left hand side (LHS), we have:

$$\frac{1}{h}\left(\mathrm{f}_q - \mathrm{f}^0_q\right) = \mathrm{K}^0_{qd}\mathrm{v}_d + \mathrm{K}^0_{qq}\mathrm{v}_q. \tag{5}$$

Our goal is to obtain zero net-force on the quasistatic nodes, so we set the force vectors to zero. (We will return to this point in §3.3.) Rearranging Eq. 5 in the form of Eq. 1, $\mathrm{v}_q = \mathrm{J}_{qd}\mathrm{v}_d$, we obtain our *con*densation *Ja*cobian (ConJac):

$$\mathrm{J}_{qd} = -(\mathrm{K}^0_{qq})^{-1}\mathrm{K}^0_{qd}. \tag{6}$$

Moving forward, we will drop the superscript 0 from K, with the understanding that these quantities are evaluated at the current time step.

3.2 Equations of Motion

Armed with the ConJac mapping in Eq. 6, we are now ready to derive the equations of motion. First, we define an expanded mapping that includes both quasistatic and dynamic nodes:

$$\mathrm{v} = \mathrm{J}\mathrm{v}_d, \quad \mathrm{v} = \begin{pmatrix}\mathrm{v}_d\\ \mathrm{v}_q\end{pmatrix}, \quad \mathrm{J} = \begin{pmatrix}\mathrm{I}\\ \mathrm{J}_{qd}\end{pmatrix}, \tag{7}$$

where I is the identity matrix. This mapping passes the dynamic velocities through untouched, while applying the ConJac mapping defined by Eq. 6 to the quasistatic velocities. Taking the time derivative of Eq. 7, we have:

$$\dot{\mathrm{v}} = \mathrm{J}\dot{\mathrm{v}}_d + \dot{\mathrm{J}}\mathrm{v}_d. \tag{8}$$

Plugging $\dot{\mathrm{v}}$ into Newton's second law, $\mathrm{M}\dot{\mathrm{v}} = \mathrm{f}$, rearranging the terms, and left multiplying by $\mathrm{J}^\top$, we get:

$$\mathrm{J}^\top\mathrm{M}\mathrm{J}\dot{\mathrm{v}}_d = \mathrm{J}^\top\left(\mathrm{f} - \mathrm{M}\dot{\mathrm{J}}\mathrm{v}_d\right). \tag{9}$$

The LHS matrix, $\mathrm{J}^\top\mathrm{M}\mathrm{J}$, is the effective inertia tensor acting on the dynamic nodes. This generalized inertia includes not only the self inertia of the dynamic nodes but also the inertia of the quasistatic nodes, since any motion of the dynamic nodes automatically causes the quasistatic nodes to move. The right hand side (RHS) vector is pre-multiplied by the Jacobian transpose, $\mathrm{J}^\top$. Since $\mathrm{J}^\top = \begin{pmatrix}\mathrm{I} & \mathrm{J}^\top_{qd}\end{pmatrix}$, the forces acting on quasistatic nodes are left-multiplied by $\mathrm{J}^\top_{qd}$ to project away the null-space. Finally, since the goal of our approach is to approximate dynamics while preserving quasistatics, we ignore the quadratic velocity vector on the RHS involving $\dot{\mathrm{J}}$, which disappears when v is zero [Shabana 2013]. In our examples, the lack of the quadratic velocity vector did not cause any visual artifacts.

The ConJac mapping can be used with a variety of time stepping schemes. In this work, we use the popular linearly implicit (which we call "Vanilla") formulation [Baraff and Witkin 1998; Lloyd et al. 2012; Müller et al. 2008; Nealen et al. 2006]. This integration scheme is easy to implement, requiring only a single linear solve per time step.

$$\left(\mathrm{M} - \beta h^2\mathrm{K}\right)\mathrm{v} = \mathrm{M}\mathrm{v}^0 + h\mathrm{f}. \tag{10}$$

Here, the tangent stiffness matrix, K, is evaluated at the current time step, but we have dropped the superscript for brevity. In addition to the h^2 factor in the stiffness term in Eq. 10, we also apply a positive factor β to control the amount of damping [Barbič and James 2005; Xu and Barbič 2017]. If we increase β, the simulation becomes more stable but at the cost of added numerical damping.

We obtain *our final* ConJac *equations of motion* by projecting Vanilla with the Jacobian:

$$\mathrm{J}^\top\left(\mathrm{M} - \beta h^2\mathrm{K}\right)\mathrm{J}\mathrm{v}_d = \mathrm{J}^\top\left(\mathrm{M}\mathrm{v}^0 + h\mathrm{f}\right). \tag{11}$$

We solve this linear system at every time step for the new dynamic velocities, v_d. Once the dynamic velocities are computed, we compute the quasistatic velocities as $\mathrm{v}_q = \mathrm{J}_{qd}\mathrm{v}_d$. Then, as explained in the next section, we apply stabilization to the positions at the end of the time step.

3.3 Stabilization

The Jacobian, J_{qd}, defined in Eq. 6 can cause large errors for nonlinear materials, due to the linear approximation introduced in Eq. 2. Since we are applying condensation at the velocity level, after taking a time step, the quasistatic forces inevitably contain small non-zero values, which implies that the LHS of Eq. 5 is not always zero. In particular, the *current* force acting on the quasistatic nodes, f_q^0, is not exactly balanced, and contains small non-zeros. (On the other hand, the implicit force at the *next* time step, f_q, is what we want to eliminate, so it is set to zero.)

This observation allows us to compute the "residual" velocity that drives the quasistatic nodes back to the zero net-force state. If we do not throw away f_q^0 from Eq. 5, we obtain:

$$\mathrm{v}_q = \mathrm{J}_{qd}\mathrm{v}_d + \mathrm{b}_q, \quad \mathrm{b}_q = -\frac{1}{h}(\mathrm{K}_{qq}^0)^{-1}\mathrm{f}_q^0. \tag{12}$$

This Baumgarte-like stabilization term, b_q, is the key term that makes our approach work, even in the presence of linearization artifacts [Baumgarte 1972]. Rather than modifying the velocities, we apply this stabilization term when we update the positions. We multiply this factor by a scalar parameter γ that controls the strength of the stabilization. The position updates for dynamic and quasistatic nodes are then:

$$\begin{aligned} \mathrm{x}_d &= \mathrm{x}_d^0 + h\mathrm{v}_d \\ \mathrm{x}_q &= \mathrm{x}_q^0 + h(\mathrm{v}_q + \gamma \mathrm{b}_q). \end{aligned} \tag{13}$$

When applied to the position, this stabilization term becomes exactly a Newton correction term: $\Delta\mathrm{x} = -\gamma\mathrm{K}_{qq}^{-1}\mathrm{f}_q$. In other words, we apply one scaled Newton step at the position level after taking a velocity step, with $\gamma = 1$ corresponding to a full Newton step. In practice, we found that a full Newton step can sometimes cause instabilities. The best value can be obtained with a line search, but we found that simply setting $\gamma = 1/3$ worked well for our examples (unless otherwise stated).

Without the stabilization term b_q, the object becomes visibly distorted due to the accumulation of error, and can eventually blow up. This term had not been derived in previous approaches because linearization does not cause any drift in linear materials. This stabilization approach is both effective and efficient. An alternative approach based on pre- or post-stabilization may work as well [Cline and Pai 2003; Weinstein et al. 2006], but we speculate that they will be less efficient and more difficult to implement.

3.4 Time Stepping

The overall simulation pseudocode for ConJac using linearly implicit Euler [Baraff and Witkin 1998] is shown in Alg. 1. For comparison, we also show the Vanilla pseudocode, also using linearly implicit integration, in Alg. 2.

With Vanilla, the performance bottleneck is the linear solve for the new velocities (line 4). On the other hand, with ConJac, solving for the new dynamic velocities is not the bottleneck because

Algorithm 1 ConJac pseudocode

```
1: (Optional) Initialize the quasistatic positions
2: Compute M
3: while simulating do
4:     Compute f, K
5:     Compute J, b
6:     Solve for v_d (Eq. 11)
7:     Compute v_q = J_qd v_d
8:     Update x (Eq. 13)
9: end while
```

Algorithm 2 Vanilla pseudocode

```
1: Compute M
2: while simulating do
3:     Compute f, K
4:     Solve for v (Eq. 10)
5:     Update x = x^0 + hv
6: end while
```

Eq. 11 is small. Instead, the bottleneck is in forming the Jacobian (line 5), which involves a series of solves by K_{qq}, which cannot be prefactored for nonlinear materials.

With our current implementation, each time step of ConJac (lines 4-8 in Alg. 1) is about 20% slower than a time step of Vanilla (lines 3-5 in Alg. 2) with 4 dynamic nodes, and 40% slower with 10 dynamic nodes (see Fig. 2). However, we more than make up for this difference because ConJac allows much bigger time steps for the same amount of dynamic behavior.

The initial nonlinear solve for the quasistatic positions in ConJac (line 1 in Alg. 1) can be costly, but it only needs to be performed once at the beginning of the simulation. We do not need to run this expensive nonlinear optimization within the simulation loop because of the stabilization term from §3.3. In fact, it is even possible to skip the initial nonlinear solve, since the stabilization term eventually eliminates the drift and drives quasistatic nodes to their zero net-force state over time.

4 RESULTS

We implemented our system in MATLAB and ran the simulations on a consumer laptop with an Intel Core i9-9880H CPU @ 2.3 GHz and 16 GB of RAM. We use MEX for filling the force vector and the stiffness matrix, and CHOLMOD for sparse linear factorizations and solves [Davis 2006]. The scene parameters are listed in Table 1. All of the objects are table-top sized—roughly 5-15 cm across, weighing a few hundred grams. For all results, we use the Stable Neo-Hookean (SNH) base material [Smith et al. 2018]. This material is stable under inversion, but like any non-linear material, it can still need a Newton solve plus line search to maintain stability under large deformations. We found that when used with a linearly implicit scheme, it must be heavily damped when using a large time step, especially when Poisson's ratio, ν, is close to 0.5.

Dragon. We start with a 10k node dragon, shown in Fig. 1. This example shows that ConJac presents an attractive option for efficiently producing lively simulations. We grab the jaws and

Table 1: List of scene parameters. #vert: number of total vertices. #dyn: number of dynamic vertices. #elem: number of elements. mat: material model. Y: Young's modulus (Pa). ν: Poisson's ratio.

Scene	#vert	#dyn	#elem	mat	Y	ν
Dragon	10456	0-10	37565	SNH	3e4	0.49
Twist	1029	1-32	4320	SNH	1e4	0.40
Hetero	5915	1	29376	SNH	1e4	0.40
Aniso	6591	1	32832	+aSTVK	1e4	0.40
Muscle	262	5	438	+aFUNG	3e4	0.49
BarCut	6050	2	29400	SNH	1e4	0.40
Bunny	5988	8	27695	SNH	4e4	0.45

the body of the dragon and pull them in different directions. After some time has passed, we let go, instantaneously releasing the built-up energy. We compare the results using ConJac and Vanilla, both with time step h =5e-3 for this 1 second simulation. For the damping factor, we use $\beta = 0.5$ for ConJac and $\beta = 3.7$ for Vanilla (Eq. 10 and Eq. 11). These values were chosen by manually searching for the smallest β values in 0.1 increments that produced stable simulations. As can be seen in the supplemental video, the discrepancy in the β values are visibly significant. *Using the same h, ConJac produces highly dynamic results, whereas Vanilla produces heavily damped results.* Since we are using the linearly implicit integrator, more dynamic results can be generated with Vanilla by reducing h, but this adds computational cost. ConJac, on the other hand, allows large time steps while retaining interesting dynamics. If we reduce the time step to h =2e-3 with Vanilla, the qualitative behavior of the dragon becomes nearly as lively as ConJac, but the wallclock simulation time increases to more than double the time of ConJac with 6 dynamic nodes. For didactic purposes, we also include a ConJac simulation with 0 dynamic nodes, which produces a quasistatic simulation driven solely by the stabilization

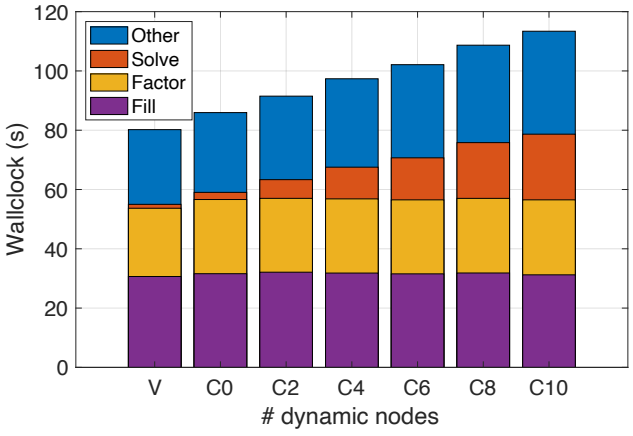


Figure 2: Wallclock times for Dragon. Starting from the left: Vanilla, ConJac with 0, 2, 4, 6, 8, and 10 dynamic nodes. Each bar is broken down into Fill (f and K), Factor, Solve, and Other. As the number of dynamic nodes increases, the Solve cost goes up linearly.

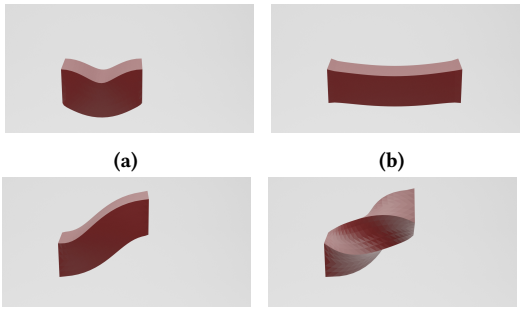
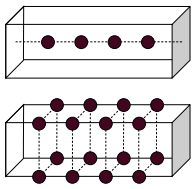


Figure 3: Twist scene: A bar is (a) compressed, (b) stretched, (c) bent, (d) and twisted through kinematic motion. The bar in these figures only have a single dynamic node.

term, b_q from Eq. 12. For this example, we used the stabilization factor $\gamma = 1/5$, since the Newton displacements immediately after releasing the jaws and the body are extremely large. Once we add dynamic nodes, the behavior becomes very lively, even with only 2 nodes. The wallclock times of ConJac is compared to Vanilla in Fig. 2. Virtually all of the added cost is in the triangular solves—since we require $3n_d+1$ solves, where n_d is the number of dynamic nodes, the cost increases linearly in n_d. (The +1 is for computing the stabilization term, b_q.) For most objects, 4 to 8 dynamic nodes are enough to produce convincingly dynamic results. We discuss potential ways to improve performance in §5.1.



Twist. Here, we show the deformation behavior of ConJac as we increase n_d, the number of dynamic nodes. For this scene, we use ConJac to simulate a bar with one of its ends moved kinematically to compress, stretch, bend, and twist the bar as shown in Fig. 3. For $n_d = \{1, 2, 4\}$, we place the dynamic nodes at equal intervals along the central horizontal axis. For $n_d = \{8, 16, 32\}$, we slice the bar orthogonal to the central axis at equal intervals and place 4 dynamic nodes at the corners of each of these vertical slices. Interestingly, it becomes difficult to visually distinguish between these cases—even with 1 dynamic node, the dynamic motion is convincing. When the dynamic nodes are placed along the central axis ($n_d = \{1, 2, 4\}$), we get the added "feature": the twisting waves are propagated instantaneously along the bar, increasing the stability of the system. If the dynamic nodes are placed along vertical slices ($n_d = \{8, 16, 32\}$), we recover the twisting dynamics.

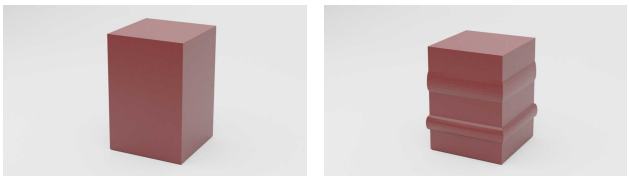
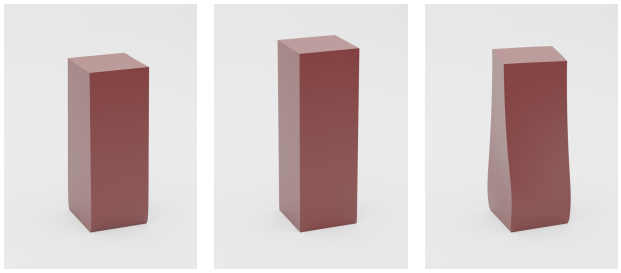


Figure 4: Hetero scene that divides a vertical bar mesh into layers of alternating material stiffness.



(a) (b) (c)

Figure 5: ANISO bar with (a) horizontal, (b) vertical, (c) helical directional fibers.

HETERO. We show that CONJAC efficiently and effectively handles heterogeneous materials. In this example, we use CONJAC to simulate a vertical bar with alternating layers of stiffnesses. Fig. 4 shows that even with only one dynamic node, we can capture the bulging of the soft layers. Because of gravity, the lower soft layer bulges out more than the upper soft layer, even though they have the same stiffness. Once the object reaches its static state, the final shape is exactly the same as the one generated by VANILLA.

ANISO. We show the effect of anisotropic materials. On top of the base SNH material, we add an anisotropic Saint Venant–Kirchhoff material (aSTVK) [Kim et al. 2019]. In this example, we use CONJAC with one dynamic node to simulate a vertical bar with different anisotropic directions: vertical, horizontal, diagonal, and helical. Fig. 5 shows that when gravity compresses the bar, it deforms differently depending on the fiber directions. Interestingly, the helical fibers induce a twisting motion.

MUSCLE. CONJAC can easily be combined with existing rigid body dynamics to model a musculoskeletal system (Fig. 6). In this 2D

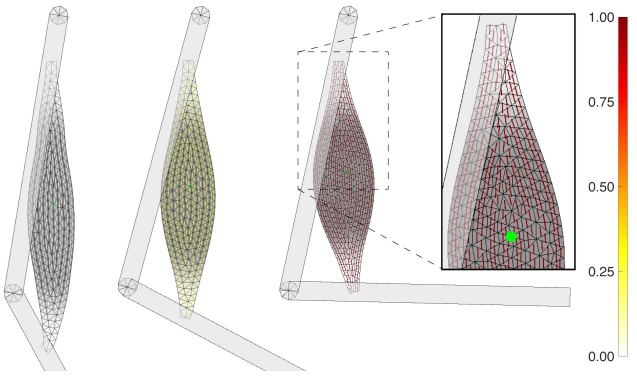
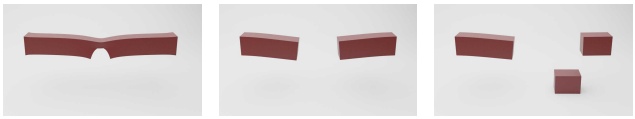


Figure 6: MUSCLE scene that combines CONJAC with rigid body dynamics. There is one dynamic node in the middle of the muscle, colored green in the inset subfigure. The colored lines in the muscle foreground show the fiber directions of the anisotropic Fung material, activated from white to yellow to red. The muscle background is color coded in gray with the stiffness of the SNH material.



(a) (b) (c)

Figure 7: BARCUT scene where a bar is stretched apart, (a) cut, and (b) fully separated into two halves. (c) The right piece is further cut into two pieces. The left and middle pieces are dynamic, while the right piece becomes quasistatic.

example, we combine CONJAC with a reduced coordinate articulated rigid body framework [Wang et al. 2019]. To attach the origin and insertion nodes to the bones, we use a Jacobian mapping that expresses the velocity of these nodes as a function of the velocities of the joints. This allows us to solve for the velocities of the muscles and joints simultaneously to give us full two-way coupling between muscles and bones, which is important because the muscle weighs more than the bones. We use SNH for the background isotropic material, and anisotropic Fung (aFUNG) for the muscle fiber material [Fung 2013]. We also take advantage of CONJAC's support for heterogeneity—the stiffness of the background SNH material is modulated so that it is stiffer in the tendon regions than in the muscle region. In the resulting simulation, the dynamics of the muscle is fully accounted for by a single, central dynamic node. In total, the system is only 4-dimensional: 2 DOFs for the joints and 2 for the muscle. Unlike quasistatic muscle simulators that assume both bones and muscles are quasistatic, with CONJAC, we can keep the bones fully dynamic and choose how dynamic we want the muscles to be.

BARCUT. In this example, we show that CONJAC supports topology changes. We start with a horizontal bar fixed at its two ends, and we cut the bar in two locations (see Fig. 7). We place two dynamic nodes on either side of the initial cut. Because CONJAC requires no precomputation, the cut can be placed anywhere. After the second cut, the right-most piece loses all dynamic nodes and gracefully degrades into a purely quasistatic model.

BUNNY. In our last example, we show that CONJAC can be extended to handle frictional contact. We drop a bunny with 8 dynamic nodes onto the floor with various starting orientations. We follow the formulation by McAdams et al. [2011] for the contact penalty force: $\mathrm{f} = K\left((1-\alpha)\mathrm{n}\mathrm{n}^{\top} + \alpha\mathrm{I}\right)(\mathrm{x} - \mathrm{x}_s)$, where K is a stiffness constant, n is the collision normal, and x_s is the closest point on the

(a) (b)

Figure 8: BUNNY falling, colliding with the floor, (a) deforming from the impact, (b) bouncing back up.

collision surface. When $\alpha = 0$, the spring acts only along the normal direction, and when $\alpha = 1$, the spring acts isotropically. In our experiments, we use $\alpha = 0.1$. For friction, we use the velocity filter approach by Bridson et al. [2002] to compute the post-friction velocity, v^f, of all nodes. For the coefficient of friction, we use a global value of $\mu = 0.3$. We then use weighted least squares to compute our new dynamic velocity: $\mathrm{v}_d^* = \mathrm{argmin}\|\mathrm{v}^f - \mathrm{J}\mathrm{v}_d\|^2_{\tilde{\mathrm{M}}}$, where $\tilde{\mathrm{M}} = \mathrm{M} - \beta h^2 \mathrm{K}$ with $\beta = 0.5$ as in other examples. This solve is inexpensive, since we solve only for the dynamic nodes of domains in contact. When collisions occur with quasistatic nodes, the contact information is added to the global stiffness matrix, making ConJac be collision-aware. ConJac intelligently transfers the masses of the quasistatic nodes to the dynamic nodes, giving us a small (24×24 in this case since there are 8 dynamic nodes) and stable system to solve at each time step. Even with only 8 dynamic nodes, ConJac gives remarkably rich deformations. For example, although the front feet and the two ears only have one dynamic node each, they undergo significant local nonlinear deformations upon contact, as shown in Fig. 8 and the supplemental video.

5 CONCLUSION

ConJac is a new reduced coordinate approach based on condensation. Unlike previous work, we apply condensation at the velocity level by defining a mapping that expresses the velocities of quasistatic DOFs as a linear function of the dynamic DOFs. Compared to Vanilla (the standard, full FE solution), ConJac remains stable at large time steps and exhibits highly dynamic motion with less numerical damping. Furthermore, ConJac gives the exact same configuration as Vanilla once the static state is reached. To demonstrate ConJac's versatility, we have shown examples involving: a wide range of materials, anisotropy and heterogeneity, topology changes, and integration with rigid body dynamics.

5.1 Limitations & Future Work

For ConJac to maintain its advantages over Vanilla, the dynamic nodes must not be too close to each other. In our Dragon and Bunny examples, we manually placed the first few dynamic nodes in strategic locations (*e.g.,* dragon jaws, bunny ears), and the rest were generated randomly. If two dynamic nodes were generated too close to each other, we reran the random generator with a different seed.

Although the stabilization term, b_q in Eq. 12, works well to fight the drift due to the linearization artifacts of the Jacobian, it still cannot maintain the zero net-force state on the quasistatic nodes during motion, causing visual artifacts especially when the motion is large. Rather than taking a single Newton step, taking multiple steps would produce better results when time steps are large. A quasi-Newton approach, where only the force vector, and not the stiffness matrix, is updated every step may yield a good balance between convergence and performance.

In our current implementation, we explicitly form J_{qd}, which requires $3n_d$ solves with K_{qq}, where n_d is the number of dynamic nodes. When n_d is small, the bottleneck is the factorization of K_{qq}, making ConJac and Vanilla nearly equivalent in terms of computational cost. As we increase n_d, the solves start to become the bottleneck, making ConJac more and more expensive compared to Vanilla. However, as shown in §4, ConJac retains important dynamics even with few dynamic nodes. An exciting avenue of future work is to follow the work of Mitchell et al. [2016] to decompose the object into domains, which would allow ConJac to scale up to a very large mesh, since then the factorizations of K_{qq} can be computed per-domain. However, obtaining good multi-threaded performance would still be a major challenge, requiring careful tuning of domain sizes and topology.

Scaling ConJac to very large meshes would require an iterative approach, since the factorization of K_{qq} may not fit into memory. This is non-trivial for the same reason above—the number of RHS vectors is $3n_d$ where n_d is the number of dynamic nodes. One approach to resolve this issue is the block Krylov method [O'Leary 1980], which allows the solver to share information across multiple RHS. However, we would still need to limit n_d to be relatively small to remain competitive.

An important limitation is that frictional impulses acting on quasistatic nodes cannot be accurately handled, since these nodes are not DOFs, and so their frictional impulses can only be satisfied in a least squares sense. This is, however, a limitation common to all reduced coordinate approaches. Therefore, our approach is most suitable when the effects of friction are not too large.

We have found experimentally that ConJac does not work well for very soft objects, due to severe linearization artifacts. For similar reasons, ConJac cannot handle extremely fast rotational motion. For these types of simulations, we may need to run Newton's method to convergence, rather than using the linearly implicit Euler scheme.

ConJac can suffer from locking artifacts with hard constraints if these constraints are applied to quasistatic nodes. In such cases, an averaged or softened constraint will need to be applied, or new dynamic nodes must be inserted [Andrews et al. 2017; Bergou et al. 2007; Tournier et al. 2015].

Finally, we are interested in exploring adaptive time step integrators, such as Runge-Kutta-Fehlberg or MATLAB's `ode45` [Fehlberg 1969; Shampine and Reichelt 1997]. Given ConJac's stability at large time steps even with an explicit integrator, these adaptive methods have the potential to reduce the number of total time steps substantially.

ACKNOWLEDGMENTS

This work was sponsored in part by the National Science Foundation (CAREER-1846368).

REFERENCES

Steven S. An, Theodore Kim, and Doug L. James. 2008. Optimizing Cubature for Efficient Integration of Subspace Deformations. *ACM Trans. Graph.* 27, 5, Article 165 (Dec. 2008), 10 pages.

Sheldon Andrews, Marek Teichmann, and Paul G. Kry. 2017. Geometric Stiffness for Real-Time Constrained Multibody Dynamics. *Computer Graphics Forum (Proc. Eurographics)* 36, 2 (May 2017), 235–246.

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

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 4 (July 2005), 982–990.

Jernej Barbič and Yili Zhao. 2011. Real-time Large-deformation Substructuring. *ACM Trans. Graph.* 30, 4, Article 91 (July 2011), 8 pages.

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

Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: Toward Directable Thin Shells. *ACM Trans. Graph.* 26, 3 (July 2007), 50–59.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Trans. Graph.* 21, 3 (July 2002), 594–603.

Morten Bro-Nielsen and Stephane Cotin. 1996. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer Graphics Forum*, Vol. 15. 57–66.

Min Gyu Choi and Hyeong-Seok Ko. 2005. Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE TVCG* 11, 1 (Jan. 2005), 91–101.

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.

Timothy A Davis. 2006. *Direct methods for sparse linear systems*. SIAM.

Erwin Fehlberg. 1969. *Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems*. Technical Report NASA-TR-R-315.

Yuan-Cheng Fung. 2013. *Biomechanics: mechanical properties of living tissues*. Springer Science & Business Media.

Ming Gao, Nathan Mitchell, and Eftychios Sifakis. 2014. Steklov-Poincaré Skinning. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 139–148.

Robert J Guyan. 1965. Reduction of stiffness and mass matrices. *AIAA journal* 3, 2 (1965), 380–380.

Bruce Irons. 1965. Structural eigenvalue problems-elimination of unwanted variables. *AIAA journal* 3, 5 (1965), 961–962.

G. Irving, J. Teran, and R. Fedkiw. 2004. Invertible Finite Elements for Robust Simulation of Large Deformation. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 131–140.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The Material Point Method for Simulating Continuum Materials. In *ACM SIGGRAPH 2016 Courses*. Article 24, 52 pages.

Theodore Kim, Fernando De Goes, and Hayley Iben. 2019. Anisotropic Elasticity for Inversion-Safety and Element Rehabilitation. *ACM Trans. Graph.* 38, 4, Article 69 (July 2019), 15 pages.

Theodore Kim and Doug L. James. 2011. Physics-based Character Skinning Using Multi-domain Subspace Deformations. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* (Vancouver, British Columbia, Canada). ACM, 63–72.

David I. W. Levin, Joshua Litven, Garrett L. Jones, Shinjiro Sueda, and Dinesh K. Pai. 2011. Eulerian Solid Simulation with Contact. *ACM Trans. Graph.* 30, 4 (July 2011), 36:1–36:10.

Minchen Li, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M. Kaufman. 2019. Decomposed Optimization Time Integrator for Large-step Elastodynamics. *ACM Trans. Graph.* 38, 4, Article 70 (July 2019), 10 pages.

Siwang Li, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao, and Mathieu Desbrun. 2014. Space-Time Editing of Elastic Motion through Material Optimization and Reduction. *ACM Trans. Graph.* 33, 4, Article 108 (July 2014), 10 pages.

John E Lloyd, Ian Stavness, and Sidney Fels. 2012. ArtiSynth: A fast interactive biomechanical modeling toolkit combining multibody and finite element simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 355–394.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans. Graph.* 30, 4, Article 37 (July 2011), 12 pages.

Nathan Mitchell, Michael Doescher, and Eftychios Sifakis. 2016. A Macroblock Optimization for Grid-based Nonlinear Elasticity. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* (Zurich, Switzerland). Eurographics Association, Goslar, Germany, 11–19.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

Matthias Müller, Jos Stam, Doug James, and Nils Thürey. 2008. Real time physics: class notes. In *ACM SIGGRAPH 2008 classes*. ACM, 88.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 21–28.

Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum* 25, 4 (2006), 809–836.

Dianne P O'Leary. 1980. The block conjugate gradient algorithm and related methods. (1980).

Zherong Pan, Hujun Bao, and Jin Huang. 2015. Subspace Dynamic Simulation Using Rotation-Strain Coordinates. *ACM Trans. Graph.* 34, 6, Article 242 (Oct. 2015), 12 pages.

Mario Paz. 1989. Modified dynamic condensation method. *Journal of Structural Engineering* 115, 1 (1989), 234–238.

A. Pentland and J. Williams. 1989. Good Vibrations: Modal Dynamics for Graphics and Animation, Vol. 23. ACM, New York, NY, USA, 207–214.

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

Lawrence F. Shampine and Mark W. Reichelt. 1997. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing* 18, 1 (1997), 1–22.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses*. Article 20, 50 pages.

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (March 2018), 15 pages.

Yun Teng, Mark Meyer, Tony DeRose, and Theodore Kim. 2015. Subspace Condensation: Full Space Adaptivity for Subspace Deformations. *ACM Trans. Graph.* 34, 4, Article 76 (July 2015), 9 pages.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. In *Computer Graphics (Proc. SIGGRAPH)*, Vol. 21. 205–214.

Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure. 2015. Stable Constrained Dynamics. *ACM Trans. Graph.* 34, 4, Article 132 (July 2015), 10 pages.

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.

Rachel Weinstein, Joseph Teran, and Ron Fedkiw. 2006. Dynamic Simulation of Articulated Rigid Bodies with Contact and Collision. *IEEE TVCG* 12, 3 (May 2006), 365–374.

Edward L Wilson. 1974. The static condensation algorithm. *Internat. J. Numer. Methods Engrg.* 8, 1 (1974), 198–203.

Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A Scalable Galerkin Multigrid Method for Real-time Simulation of Deformable Objects. *ACM Trans. Graph.* 38, 6, Article 162 (Nov. 2019), 162:1–162:13 pages.

Hongyi Xu and Jernej Barbič. 2017. Example-Based Damping Design. *ACM Trans. Graph.* 36, 4, Article 53 (July 2017), 14 pages.

Juyong Zhang, Yue Peng, Wenqing Ouyang, and Bailin Deng. 2019. Accelerating ADMM for efficient simulation and optimization. *ACM Trans. Graph.* 38, 6, Article 163 (Nov. 2019), 163:1–162:21 pages.