# Skipping Steps in Deformable Simulation with Online Model Reduction

Theodore Kim[1,2] Doug L. James[2]
University of Saskatchewan[1] Cornell University[2]


**Figure 1:** *We learn fast subspace models on-the-fly to accelerate deformable simulation: (Top) Ground truth frames from a 1000-frame deformable character simulation with 165,941 tetrahedra and Arruda-Boyce material which took 48 hours (174,521 seconds). (Middle) Frames from our online integrator which were generated in only 51 minutes (3,101 seconds) – $\mathbf{56.28}\times$ faster – by learning a fast 11-dimensional subspace model on-the-fly (Bottom) Visualization of the temporal behavior of the online integrator reveal full steps (in black), reduced steps (in gray), and basis updates (red ticks). A reduced basis is detected quickly, and the first "skip" occurs at timestep 2. Only a handful of full steps are needed, and our reduced solver computes over 98% of the the steps.*

## Abstract

Finite element simulations of nonlinear deformable models are computationally costly, routinely taking hours or days to compute the motion of detailed meshes. Dimensional model reduction can make simulations orders of magnitude faster, but is unsuitable for general deformable body simulations because it requires expensive precomputations, and it can suppress motion that lies outside the span of a pre-specified low-rank basis. We present an online model reduction method that does not have these limitations. In lieu of precomputation, we analyze the motion of the full model as the simulation progresses, incrementally building a reduced-order nonlinear model, and detecting when our reduced model is capable of performing the next timestep. For these subspace steps, full-model computation is "skipped" and replaced with a very fast (on the order of milliseconds) reduced order step. We present algorithms for both dynamic and quasistatic simulations, and a "throttle" parameter that allows a user to trade off between faster, approximate previews and slower, more conservative results. For detailed meshes undergoing low-rank motion, we have observed speedups of over an order of magnitude with our method.

**CR Categories:** I.6.8 [Simulation and Modeling]: Types of Simulation—Animation, I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

**Keywords:** Dimensional model reduction; reduced-order modeling; subspace integration; subspace deformation; nonlinear solid mechanics; character skinning

## 1 Introduction

Physically based deformable models in offline computer animation have seen a dramatic increase in geometric complexity in recent years due to the demand for increased realism, e.g., in character animation. However, despite increases in computing power, animations of detailed meshes capable of capturing the features of a character's flesh can still take hours or days to simulate.

Model reduction (also known as subspace methods) can accelerate

the simulation of detailed deformable models by several orders of magnitude, but their impact on offline physically based animation has so far been limited for two main reasons. First, model reduction achieves superior runtime performance only after exploiting an extensive precomputation stage. In the offline case, a simulation is only computed once, so the cost of the precomputation nullifies any speed gains obtained during the actual simulation. Second, model reduction limits the motion of the model to a pre-specified basis, and any attempted motion outside this basis is artificially suppressed. Unfortunately, the minimum subspace necessary to faithfully capture the configurations space of an offline animation is not generally known ahead of time. While intelligent guesses are possible, the likelihood of some interesting motion being suppressed increases as an animation becomes more complex. Since the main advantage of physically based techniques is their ability to automatically capture novel dynamics, such suppression is unacceptable.

However, principal component analysis (PCA) of simulation results often reveals that detailed meshes in fact undergo very low-rank motion. This gives rise to the question: is it possible to somehow discover this low-rank subspace "on the fly," and exploit it to make the simulation faster? In this paper, we design an online subspace integrator that achieves this goal. We factor snapshots of the full-model simulation state immediately after they are computed and incrementally build a reduced nonlinear model. We then use a variety of error estimators to predict if the next timestep can be performed by our current reduced model, and if so, replace the full-model computation with a very inexpensive (on the order of milliseconds) reduced order step. We have used our method to successfully discover the low-rank motion of a variety of simulations and subsequently accelerate them by over an order of magnitude. Note "skipping steps" can in principle accelerate simulations which already have complex solvers, e.g., multigrid.

We allow the user to directly control the tradeoff between the speed and quality of a simulation by providing a "throttle" parameter. When the throttle is set high, the user can obtain fast, approximate previews of the simulation motion without having to resort to lower resolution meshes that may not faithfully represent high resolution features. Once the user is satisfied with the previews, a more time-consuming full simulation can be launched. If the throttle is set low, the integrator will only use the reduced model when it is very confident that the amount of error being introduced is low. This will still provide some speedup over the full simulation alone.

## 2 Related Work

Physically based simulation of deformable bodies has been an active topic research in computer graphics for over two decades [Terzopoulos et al. 1987; Terzopoulos and Witkin 1988; Terzopoulos and Fleischer 1988], has experienced great success in simulating such phenomena as cloth [Baraff and Witkin 1998; Bridson et al. 2002] and fracture [O'Brien and Hodgins 1999; Molino et al. 2004], and has recently made great strides in robustness [Irving et al. 2004; Teran et al. 2005]. While we briefly summarize related work here, excellent existing surveys [Gibson and Mirtich 1997; Nealen et al. 2005] provide more complete treatments.

Our current work is orthogonal and complementary to hierarchical, multi-resolution methods [Debunne et al. 2001; Capell et al. 2002; Grinspun et al. 2002]. Hierarchical methods accelerate simulations by detecting and exploiting spatially local coherence, while our method exploits coherence in the global motion of a model. A multi-resolution solver can play a crucial role in accelerating the unreduced simulation and expediting the arrival of example data to the subspace integrator. However, the running time of a multi-resolution solver is inherently $N$-dependent, where $N$ is the mesh size. A subspace integrator constructs an $O(r)$ subspace where $r \ll N$. When a suitable subspace is found, the subsequent $r$-dependent timesteps contribute significant additional speedup.

Most model reduction methods are *a posteriori*, because they first construct a fixed basis by analyzing a static model (e.g. using linear model analysis (LMA) [Shabana 1990]) or a set of simulation "snapshots" (e.g. using principal component analysis (PCA) [Krysl et al. 2001]). Various extensions exist for LMA and PCA, such as modal derivatives [Idelsohn and Cardona 1985] for the former, and "interactive sketching" as a data collection strategy for the latter [Barbič and James 2005]. Recent reduced-order methods for fluids in graphics [Treuille et al. 2006; Wicke et al. 2009] are also *a posteriori*, as they perform PCA on a set of fluids snapshots.

We are instead interested in *a priori* model reduction, which is not restricted to any initial basis. A Priori Hyper-Reduction (APHR) [Ryckelynck 2005] recently demonstrated the viability of such an approach to finite element method (FEM) simulations of non-linear thermal conduction. APHR has subsequently been extended to boundary element method (BEM) simulations [Ryckelynck et al. 2005], as well as viscoelastic and viscoplastic FEM simulations [Ryckelynck 2009]. APHR performs subspace simulation steps while periodically checking the magnitude of the full residual. If the residual is too large, the basis is enriched with Krylov vectors, the simulation is backed up several timesteps, and then restarted. Such backups are not acceptable in offline animation, since they would require coupled simulators (fluids, rigid bodies) to be backed up as well, which would invalidate any performance gains. The dual weighted residual (DWR) method [Meyer and Matthies 2003] as well as the method of Homescu et al. [2006] take a different approach and solve the adjoint of a known simulation result backwards in time to obtain *a priori* reduced-order error bounds on perturbed versions of the simulation. These two methods still bootstrap off of existing simulation data, so they cannot be applied to the case where the simulation is only computed once. Utku et al. [1985] proposed an *a priori* method of estimating the reduced-order error of a non-linear system over time with the goal of evaluating the effectiveness of different bases. While the details differ significantly, our error estimator for dynamic simulation shares the same high-level strategy of linearizing about the last known unreduced state.

Our online subspace integrator is motivated by the fact that many animations are often much lower rank than their underlying models. For example, pose-space deformation [Lewis et al. 2000] successfully constructs low-dimensional approximations to posed high-rank meshes by using scattered data interpolation. Wang and Phillips [2002] estimated "multi-weight enveloping" approximations to offline deformable simulations to accelerate complex character models. Similarly, *Key Point Subspace Acceleration (KPSA)* [Meyer and Anderson 2007] constructs a low-rank approximation to a set of training poses, and provides a set of "key points" for an animator to pose. A similar notion was used to select a set of "key tets" as cubature points to approximate the force and stiffness response of a deforming model [An et al. 2008], and the number of key tets necessary was found to grow linearly with the rank of the subspace. The EigenSkin algorithm [Kry et al. 2002] constructed a sparse basis from offline FEM snapshots to correct the results of a skinned model, and found that even a small (rank 1) basis resulted in a significant reduction in skinning error. James and Twigg [2005] demonstrated that clustering and least-squares approximation could be used to detect bones and bone weights, and automatically construct low-rank skinning approximations to existing mesh animations. The possibility of efficient inter-object collision handling between reduced deformable models has also been demonstrated [James and Pai 2004; Barbič and James 2007]. *Unfortunately, none of these methods can adapt their reduced-order models on-the-fly to accelerate general-purpose deformable simu-*

*lations.*

# 3 Subspace Deformation Basics

## 3.1 Equations of Motion

We write the equations of motion for a general deformable body as:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}(\mathbf{u}) = -\mathbf{f}_{ext}, \tag{1}$$

where $\mathbf{u}$ is a displacement vector, $\mathbf{M}$ is a mass matrix, $\mathbf{C}$ is a damping matrix, $\mathbf{f}_{ext}$ are external forces, and $\mathbf{f}(\mathbf{u})$ is an internal force response. We will use uppercase to denote a matrix, lowercase to denote a vector, and overdots to denote differentiation with respect to time. For a deformable model with $N$ degrees of freedom, $\mathbf{u} \in \mathbb{R}^N$, $\mathbf{f}_{ext} \in \mathbb{R}^N$, $\mathbf{M} \in \mathbb{R}^{N\times N}$, $\mathbf{C} \in \mathbb{R}^{N\times N}$, and $\mathbf{f}(\mathbf{u}) \in \mathbb{R}^N$. We use standard Rayleigh damping, where $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}_0$. $\mathbf{K}_0 \in \mathbb{R}^{N\times N}$ and denotes the stiffness matrix at the rest pose. We integrate this system forward through time using a Newton-Raphson solver, with a preconditioned conjugate gradient (PCG) solver invoked at every iteration to solve the $N \times N$ system.

## 3.2 Quasistatic Deformation

For applications such as character animation, quasistatic simulation can be preferred over dynamics simulation because it provides less history-dependent behavior. While Eqn. 1 describes dynamic simulation, quasistatics assume that velocity and acceleration ($\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$) are negligible and that the model has reached an equilibrium state. Therefore, Eqn. 1 becomes a simpler equation: $\mathbf{f}(\mathbf{u}) = -\mathbf{f}_{ext}$. The displacements $\mathbf{u}$ is solved for again using Newton-Raphson and PCG. Specifically, the force balance equation

$$\mathbf{K}(\mathbf{u})\,\Delta\mathbf{u} = -\mathbf{f}_{ext} - \mathbf{f}(\mathbf{u}) \tag{2}$$

is formed, where $\mathbf{K}(\mathbf{u})$ is the stiffness matrix at the displacement $\mathbf{u}$. Newton-Raphson solves for $\mathbf{u}$ by using PCG to repeatedly compute the correction term $\Delta\mathbf{u}$. The force residual $\mathbf{f_r}$ is the current difference between the sides of Eqn. 2,

$$\mathbf{f_r} = \mathbf{K}(\mathbf{u})\,\Delta\mathbf{u} + \mathbf{f}_{ext} + \mathbf{f}(\mathbf{u}), \tag{3}$$

and characterizes how far from a fully balanced solution the current $\mathbf{u}$ is. If the norm of a force residual is large, it indicates that the current $\mathbf{u}$ is far from the equilibrium solution. We will use this fact later when designing our quasistatic error estimator.

## 3.3 Reduced Equations of Motion

Subspace deformation techniques reduce the runtime cost of integrating the equations of motion by approximating the large $N \times N$ system with a much smaller $r \times r$ system. This is accomplished by systematically applying a basis matrix $\mathbf{U} \in \mathbb{R}^{N\times r}$ to Eqn. 1. Assuming the columns of $\mathbf{U}$ are orthogonal, the mass matrix and damping matrices are projected down to their reduced equivalents $\widetilde{\mathbf{M}} = \mathbf{U}^T\mathbf{M}\mathbf{U}$ and $\widetilde{\mathbf{C}} = \mathbf{U}^T\mathbf{C}\mathbf{U}$. Vector quantities such as $\mathbf{f}_{ext}$, as well as the displacement $\mathbf{u}$ and its derivatives, are also approximated by the reduced displacements, e.g. $\mathbf{q} = \mathbf{U}^T\mathbf{u}$. The reduced equations of motion are then

$$\widetilde{\mathbf{M}}\ddot{\mathbf{q}} + \widetilde{\mathbf{C}}\dot{\mathbf{q}} + \widetilde{\mathbf{f}}(\mathbf{q}) = -\widetilde{\mathbf{f}}_{ext}. \tag{4}$$

Newton-Raphson and PCG are instead performed on this smaller system. However, while the values of $\widetilde{\mathbf{M}}$ and $\widetilde{\mathbf{C}}$ can be precomputed, the internal force $\widetilde{\mathbf{f}}(\mathbf{q})$ varies depending on the current model configuration, and cannot be precomputed.

One option is to instead compute $\mathbf{f}(\mathbf{u})$ and subsequently $\widetilde{\mathbf{f}}(\mathbf{u}) = \mathbf{U}^T\mathbf{f}(\mathbf{U}\mathbf{q})$ at every iteration [Krysl et al. 2001]. However, computing $\mathbf{f}(\mathbf{u})$ introduces an $N$-dependent cost, and compromises the goal of constructing an $r$-dependent algorithm that integrates entirely inside the subspace. In order to avoid this dependence, we approximate both $\widetilde{\mathbf{f}}(\mathbf{q})$ and its Jacobian $\widetilde{\mathbf{K}}(\mathbf{q})$ with a cubature scheme [An et al. 2008]. Briefly, given the strain energy $\Psi(X;\mathbf{q})$ of a material in reduced coordinates $\mathbf{q}$, the reduced force $\widetilde{\mathbf{f}}(\mathbf{q})$ over the mesh domain $\Omega$ is the integral:

$$\widetilde{\mathbf{f}}(\mathbf{q}) = -\int_\Omega \nabla_{\mathbf{q}}\Psi(X;\mathbf{q})\,d\Omega_X. \tag{5}$$

The gradient $\nabla_{\mathbf{q}}\Psi(\mathbf{q})$ denotes the reduced first Piola-Kirchoff stress. The cubature scheme approximates this integral with a discrete set of $n$ carefully chosen "key tets":

$$\widetilde{\mathbf{f}}(\mathbf{q}) = -\int_\Omega \nabla_{\mathbf{q}}\Psi(X;\mathbf{q})\,d\Omega_X \approx -\sum_{i=1}^{n} w_i\nabla_{\mathbf{q}}\Psi(X_i;\mathbf{q}). \tag{6}$$

The $X_i$ term denotes the key-tet material position, and $w_i$ the cubature weight of some tet $i$. The scheme essentially computes a weighted sum of first-Piola Kirchoff stresses over a set of key tets, and the negated sum is projected into the subspace $\mathbf{U}$. The reduced stiffness matrix $\widetilde{\mathbf{K}}(\mathbf{q})$ is evaluated analogously. If $O(r)$ key tets are used, then the overall algorithm remains $r$-dependent. The algorithmic detail to note is that the key tets and weights are dependent on $\mathbf{U}$, so if we incrementally construct $\mathbf{U}$ in an online manner, we must incrementally update the cubature scheme as well.

# 4 Online Model Reduction

In this section, we will often refer to the "full solver" or a "full step", and the "reduced solver" or a "reduced step". In all of these cases, "full" refers to the unreduced, full-rank solver with $N$-dependent cost, and "reduced" refers to the incrementally built, low-rank reduced-order solver with some $r$-dependent ($N$-independent) cost.

## 4.1 Basis Updating and Downdating

The sources of error in dynamic and quasistatic simulations differ significantly, so we have found it necessary to design separate online algorithms for each. However, both algorithms construct their reduced models in the same way, so we will begin by describing this shared component.

**Updating U:** We initialize the basis matrix to be empty, $\mathbf{U} \in \mathbb{R}^{N\times 0}$, and build it incrementally by updating a QR factorization on simulation results as soon as they are produced by the full solver. When a vector $\mathbf{u}$ arrives, it is orthogonalized against the existing columns in $\mathbf{U}$ using modified Gramm-Schmidt [Golub and Van Loan 1996]. If its relative RMS-norm after orthogonalization is larger than some threshold $\tau_{discard}$, then it is concatenated to $\mathbf{U}$. We compute the RMS norm as $\|\mathbf{u}\|_{RMS} = \frac{\|\mathbf{u}\|_2}{|\mathbf{u}|}$, where $|\mathbf{u}|$ is the cardinality of $\mathbf{u}$. This process is summarized in lines 4-8 of Algorithm 1.

**Downdating U:** Model reduction is most effective when the subspace rank satisfies $r \ll N$, so the number of columns in $\mathbf{U}$ should be limited to some maximum rank $r_{max}$. A simulation will almost certainly take more than $r_{max}$ timesteps, so a downdating strategy is needed for when $\mathbf{U}$ already contains $r_{max}$ columns and the full simulation provides a significant new $\mathbf{u}$ vector. Ideally, we would only discard the portions of the basis that will never be encountered again, so we base our downdating strategy on the assumption that

**Algorithm 1**: modifyBasis($\mathbf{v}$, $\mathbf{U}$)

**Data**: Candidate snapshot vector, $\mathbf{u}$; Current basis $\mathbf{U}$

**begin**
  **if** $r \geq r_{max}$ **then**
    downdateBasis()
  $\mathbf{u}_\perp \leftarrow$ make $\mathbf{u}$ perpendicular to $\mathbf{U}$
  **if** $\frac{\|\mathbf{u}_\perp\|_{RMS}}{\|\mathbf{u}\|_{RMS}} > \tau_{discard}$ **then**
    $\mathbf{u}_\perp = \mathbf{u}_\perp / \|\mathbf{u}_\perp\|_2$
    $\mathbf{U} \leftarrow [\mathbf{U}\ \mathbf{u}_\perp]$
    $r$++
    Record new $[\mathbf{r}(\mathbf{u}), \mathbf{u}]$ cubature training pair
    Retrain cubature
    Delete all key tets $i$ where $w_i = 0$
  return $\|\mathbf{u}_\perp\|_{RMS}$
**end**

the most recently seen states are the best predictor for future states. If we are downdating at timestep $t$, we concatenate the most recent reduced coordinates into a matrix $\mathbf{A} = \left[\mathbf{q}_{t-1}\mathbf{q}_{t-2}...\mathbf{q}_{t-r_{max}/2}\right]$ and compute its SVD, $\mathbf{A} = \mathbf{R}\Sigma\mathbf{V}^T$. We then use $\mathbf{R}$ to apply a rotation to the basis $\mathbf{U}$ so that the most important directions from the last $\frac{r_{max}}{2}$ reduced states are preserved. The new downdated basis is $\mathbf{U}_{new} = \mathbf{U}\mathbf{R}$. Computing the SVD of $\mathbf{A}$ is $O(r^3)$, but since we are typically dealing with $r \approx 30$, this cost is negligible. This process is summarized in lines 2-6 of Algorithm 2.

**Algorithm 2**: downdateBasis()

**Data**: Current basis $\mathbf{U}$; $\mathbf{A} = [\mathbf{q}_{t-1}\ \mathbf{q}_{t-2}\ \ldots\ \mathbf{q}_{t-r_{max}/2}]$

**begin**
  $\mathbf{U}_{old} \leftarrow \mathbf{U}$
  $r = \lfloor r_{max}/2 \rfloor$
  $\mathbf{R}\Sigma\mathbf{V}^T \leftarrow$ compute SVD of $\mathbf{A}$
  $\mathbf{U} \leftarrow \mathbf{U}_{old}\mathbf{R}$
  discard cubature training data, and $\mathbf{A}$ cache
**end**

**Updating Cubature:** The cubature scheme is basis dependent, so when $\mathbf{U}$ is updated or downdated, the key tets and weights in Eqn. 6 must be modified as well. In order to compute the weights $w_i$, the cubature optimizer [An et al. 2008] requires example force data, which we obtain by recording the corresponding force snapshot $\mathbf{f}(\mathbf{u})$ whenever a snapshot $\mathbf{u}$ is added to $\mathbf{U}$. Upon updating, the new $\mathbf{U}$, all the recorded force snapshots, and the previous set of key tets are sent to the cubature optimizer. When the optimization completes, we discard all the key tets with weights set to zero. Since the optimizer is randomized and greedy, this provides a way of discarding inferior key tets if new ones are found that provide a better fit. When a downdate occurs, there is no longer any guarantee that $\mathbf{U}$ will be able to reproduce the deformations corresponding to the force snapshots, so we delete all the force snapshots.

## 4.2 Error Estimation

The dynamic and quasistatic online integrators both use Algorithm 1 to construct their reduced order models. However, a method is still needed to determine if the reduced model can be used to take the next timestep in place of the full integrator, or if doing so will introduce an unacceptable amount of error. Such an error estimator must obviously avoid taking a full integrator timestep, but it must also be inexpensive, ideally $r$-dependent, because otherwise the cost of computing the estimate could invalidate the performance gains of the reduced model. The sources of error are sufficiently different between the dynamic and quasistatic cases that we have designed separate estimators for each.

### 4.2.1 Quasistatic Error Estimation

The main source of error in the quasistatic case is the inability of the basis to resolve the mesh pose at the equilibrium solution. As noted in Section 3.2, any deviation from equilibrium will produce a non-zero force residual, and large deviations will produce large residuals. Therefore, if we step the reduced integrator, compute $\mathbf{u} \leftarrow \mathbf{U}\mathbf{q}$, $\mathbf{f_r}$, and $\|\mathbf{f_r}\|$, and find that the norm is large, we know that the error is large and the full solver should be used. However, computing $\mathbf{f_r}$ is an operation with $N$-dependent cost, and the definition of a "small" $\|\mathbf{f_r}\|$ is material dependent. These two problems are addressed below.

While computing the full norm $\|\mathbf{f_r}\|$ takes $N$-dependent work, the norm can be efficiently approximated via sampling. For any mesh vertex, we can obtain its one-ring tetrahedra and use them to compute the vertex entries for $\mathbf{f_r}$ in $O(1)$ time. If we do this for $O(r)$ random mesh points, we can obtain a highly satisfactory approximation to $\|\mathbf{f_r}\|$ in $O(r^2)$ time. This process is summarized in Algorithm 3.

**Algorithm 3**: estimateForceResidual()

**begin**
  $\mathsf{S} \leftarrow$ select sample points
  **for** $i \in \mathsf{S}$ **do**
    Identify vertex $i$ and one-ring tetrahedra
    $\mathbf{U}_i \leftarrow$ retrieve basis rows for vertex $i$
    $\mathbf{f_{est}}_i \leftarrow \mathbf{K}(\mathbf{U}_i\mathbf{q})_i\mathbf{U}_i\mathbf{q} + \mathbf{f_{ext}}_i + \mathbf{r}(\mathbf{U}_i\mathbf{q})$
  return $\|\mathbf{f_{est}}\|$
**end**

The definition of a "small" residual norm depends on the material model. For example, given the same displacement error, a stiff material would produce a much larger residual than a soft material. The correspondence can be resolved by solving Eqn. 2, but this is precisely the computation we are trying to avoid. Instead, we can learn an upper bound on what constitutes a "small" residual norm for a specific material as the simulation progresses. When a full step is taken, we also take a reduced step and compute the true error, $\mathbf{err}$. If the $\|\mathbf{err}\|_{RMS} < \tau_{error}$, and its corresponding residual $\|\mathbf{f_r}\|_{RMS}$ is larger than any norm less than $\tau_{error}$ seen so far, then $\|\mathbf{f_r}\|_{RMS}$ is stored as the upper bound $\tau_{upper}$. The next timestep, the reduced model is stepped, and an estimate to $\|\mathbf{f_r}\|_{RMS}$, $f_{est}$, is computed. If the estimate is larger than $\tau_{upper}$, the error is predicted to be greater than $\tau_{error}$, and a full step is taken. Otherwise, the reduced result is used to skip the full step. The complete quasistatic online integrator can now be summarized in Algorithm 4. The throttle parameter, $\mathsf{T}_{quasi}$ will be defined in section 4.3.

### 4.2.2 Dynamic Error Estimation

In the quasistatic case, error occurred when the basis could not resolve the equilibrium pose of the mesh. The simulation could recover from such error in a single step, because if a full step was subsequently taken, the correct pose could still be retrieved. In dynamics however, the error is history dependent. If a run of consecutive reduced steps were taken that erroneously dissipated away important velocities and accelerations, taking a full step at the end of the run would not recover these quantities. Unless we rewound the simulation, the information would be irrecoverable, and the simulation

**Algorithm 4**: stepOnlineQuasistatic()

```
1  begin
2      f_est ← 2 × τ_upper
3      if r > 0 then
4          q ← take subspace timestep
5          f_est ← estimateForceResidual()
6      if f_est > T_quasi · τ_upper then
7          u_old ← u
8          u ← take full timestep
9          modifyBasis(u, U)
10         q ← U^T u_old
11         q ← take subspace timestep
12         err ← u − Uq
13         if ||err||_RMS < τ_error then
14             f_r ← K(Uq)Uq + f_ext + r(Uq)
15             if ||f_r||_RMS > τ_upper then
16                 τ_upper = ||f_r||_RMS
17     else
18         u ← Uq
19     A ← [Aq]
20 end
```

motion would be corrupted. We found that the force residual-based estimator from the previous section detects error only after it has become irrecoverable, and a more conservative approach is needed for dynamics. We used two complementary error estimators.

**Estimating how many steps to skip:** The first error estimator predicts how many consecutive reduced timesteps can be taken based on the orthogonal component of the last full step. This component, $\|\mathbf{u}_\perp\|_{RMS}$ is already computed during Algorithm 1, and is returned at the end of that process. The maximum number of consecutive reduced steps that can be taken is estimated to be

$$s_{max} = \left\lfloor \frac{\tau_{discard}}{\|\mathbf{u}_\perp\|_{RMS}} \right\rfloor. \tag{7}$$

We essentially assume that the previous error $\|\mathbf{u}_\perp\|_{RMS}$ will compound linearly with each step, and estimate how many steps would elapse before the error became so large that it would be added as a new column to $\mathbf{U}$. This estimate is conservative – subspace error is usually characterized as a sum of projection error and integration error (for example, see Homescu et al. [2006]), with integration error being the only component that compounds over time. We pessimistically assume that all of the error is integration error.

**Estimating subspace error:** The second error estimator predicts what the next column will be that gets added to $\mathbf{U}$, and concatenates it temporarily as an "error column," $\mathbf{u}_{err}$, to the end of $\mathbf{U}$. If its corresponding entry in $\mathbf{q}$, $q_{err}$, begins to grow, then we know that the other columns in $\mathbf{U}$ are insufficient to capture the current motion, and a full step should be taken. Notably, $\mathbf{u}_{err}$ and $q_{err}$ capture the information that otherwise would have been irrecoverable, so when the full step is taken, the motion of the model will have been minimally corrupted.

The selection of the error column $\mathbf{u}_{err}$ is critical. We chose to compute $\mathbf{u}_{err}$ using the Hessian $\mathbf{H}(\mathbf{u})$ about the current $\mathbf{u}$. $\mathbf{H}(\mathbf{u})$ is the Jacobian of the stiffness matrix $\mathbf{K}(\mathbf{u})$ and thus characterizes how $\mathbf{K}(\mathbf{u})$ will change as the mesh moves in some direction away from the current $\mathbf{u}$. We assume that the mesh will continue to move along its current trajectory, $\dot{\mathbf{u}}$, and thus choose to contract $\mathbf{H}(\mathbf{u})$ twice with $\dot{\mathbf{u}}$. The resulting vector $\Delta\mathbf{f}(\mathbf{u})$ characterizes new forces that the mesh will experience as it moves along $\dot{\mathbf{u}}$. The new displacement $\Delta\mathbf{u}$ induced by $\Delta\mathbf{f}(\mathbf{u})$ is then solved for and used as $\mathbf{v}_{err}$. This process is summarized in Algorithm 5.

We have found that by using this combination of error estimators, it is possible to perform dynamic online model reduction simulations without significantly corrupting the motion of the model. The complete algorithm is described in Algorithm 6.

### 4.3 Throttle Parameters

In both the quasistatic and dynamic case, we provide throttle parameters, $\mathsf{T}_{quasi}$ and $\mathsf{T}_{dyn}$, which are respectively on lines 6 and 9 of Algorithms 4 and 6. The intuition behind both are the same: larger values tell the online integrator to be more optimistic and take more reduced steps than it otherwise would, whereas smaller values force the integrator to be more conservative. The default value for both are 1, and a setting of 0 effectively deactivates the reduced solver. Further settings will be discussed along with the results.

**Algorithm 5**: computeErrorVector($\mathbf{u}$)

```
   Data: Current state u
1  begin
2      Δf(u) ← (H(u) : u̇)u̇
3      A ← system matrix from last full step
4      Solve AΔu = Δf(u)
5      return Δu
6  end
```

**Algorithm 6**: stepOnlineDynamic()

```
   Data: full stores whether to take a full or reduced step, initialized
         to true; s_max, the maximum consecutive reduced steps that
         can be taken, initialized to 0; s_current, number of
         consecutive reduced steps taken so far, initialized to 0.
1  begin
2      if full then
3          [u u̇ ü] ← take full timestep
4          if r > 0 then
5              [q q̇ q̈] ← take subspace timestep
6              err ← u̇ − Uq̇
7          ||u_⊥||_RMS = modifyBasis(u, U)
8          if ||err||_RMS/||u̇||_RMS < τ_error then
9              s_max = T_dyn · ⌊τ_discard/||u_⊥||_RMS⌋
10             if s_max > 0 then
11                 u_err ← computeErrorVector(u)
12                 modifyBasis(u_err, U)
13                 s_current ← 0
14                 full ← false
15         [q q̇ q̈] ← U^T[u u̇ ü]
16     else
17         [q q̇ q̈] ← take subspace timestep
18         [u u̇ ü] ← U[q q̇ q̈]    // Sometimes optional
19         s_current + +
20         if (q²_err/N)^½ > τ_error || s_current == s_max then
21             delete v_err from U
22             full ← true
23     A ← [Aq]
24 end
```

## 5 Results

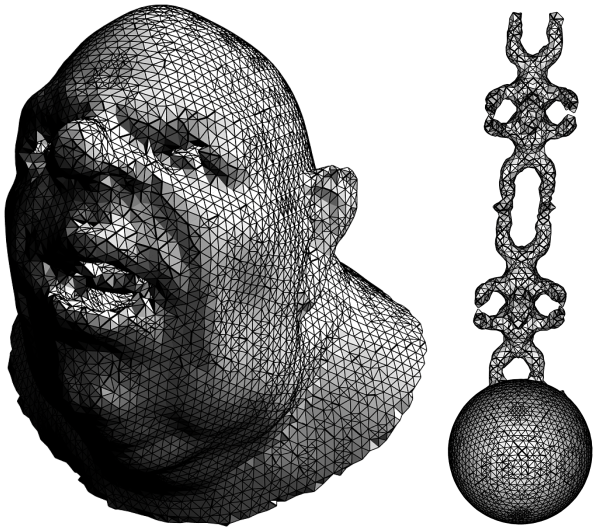

**Figure 2:** *Simulation meshes used in our examples.* **Left:** *Head mesh used for Fig. 1, containing 165,941 tetrahedra (inside is hollow). An embedded high resolution surface mesh containing 1,420,970 vertices and 2,835,314 triangles was used for the final renders.* **Right:** *Pendulum mesh used for Fig. 4 containing 64,275 tetrahedra. No high resolution mesh was used for the final renders.*

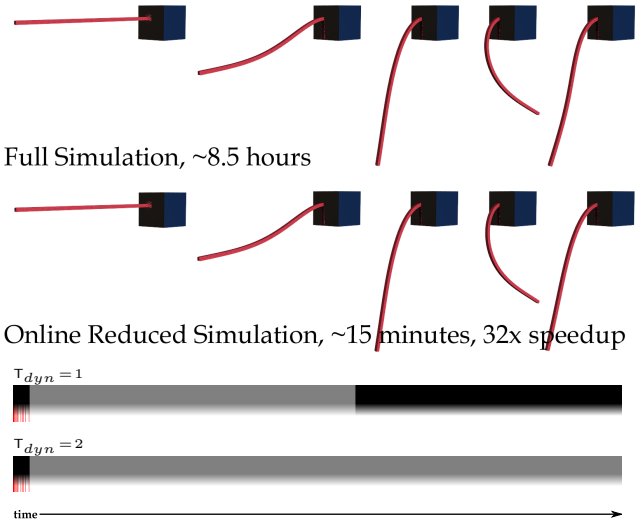


**Figure 3:** *Frames from a 1500 step cantilever beam simulation. The online integrator is up to* $\mathbf{32.58}\times$ *faster than the full simulation. Along the bottom is a visualization of the integrator steps, with the same coloring as Fig. 1. The discovered bases are identical, but the default setting of* $\mathsf{T}_{dyn}=1$ *conservatively skips only 53.5% of the steps. The user can specify a more aggressive* $\mathsf{T}_{dyn}=2$*, resulting in 97.27% of the steps being skipped.*

We describe numerical experiments with the online integrator for simulations of two dynamic models and one quasistatic model. In all cases, the online integrator automatically constructed a low-rank reduced-order model and successfully skipped a large number of steps. All of the following timings were obtained on an 8-core 3.0 Ghz Intel Xeon with 8 GB of RAM. The full solver used the PCG solver in PetSc [Balay et al. 2001], with Intel's Math Kernel Library (MKL) handling the LAPACK and BLAS routines. The reduced solver used the Intel MKL for BLAS calls and Cholesky solves. OpenMP was used to accelerate stiffness matrix construction in both solvers and key-tet selection in the cubature optimizer. The underlying tetrahedral models used in our simulations can be seen in Fig. 2. For the head example (Fig. 1), an embedded high resolution mesh was used for the final renders. The mesh for the beam example (Fig. 3) is not shown because it is just a regular line of tetrahedra. All of the meshes were generating using IsoSurface Stuffing [Labelle and Shewchuk 2007].

The first dynamic model is a thin, cantilever beam composed of 16,824 tets, St. Venant-Kirchhoff (StVK) material with $\lambda = 20000$ and $\mu = 100000$, and a gravity force applied (Figure 3). While many graphics papers use a thick beam to demonstrate the efficacy of a deformation method (see e.g. [Barbič and James 2005]), the beam usually does not exhibit significant dynamics, and would have been a trivial case for our online integrator. We instead chose a more challenging case where the beam bounces and swings for most of the simulation. Even with these more complex dynamics, the online integrator discovered a rank 13 subspace containing 112 key tets and skipped 802 to 1500 (53.47%) timesteps for $\mathsf{T}_{dyn} = 1$ and 1459 of 1500 (97.27%) timesteps for $\mathsf{T}_{dyn} = 2$. Inside the online integrator, the full integrator averaged 18.5 seconds per timestep, while the reduced integrator averaged 88 milliseconds for $\mathsf{T}_{dyn} = 1$, and respectively 21.31 seconds and 46 ms for $\mathsf{T}_{dyn} = 2$. A full simulation completed in 30,658 seconds, and the online integrator completed in 12,987 ($\mathsf{T}_{dyn} = 1$) and 941.01 ($\mathsf{T}_{dyn} = 2$) seconds, giving speedups of $\mathbf{2.36}\times$ and $\mathbf{32.58}\times$.

The second dynamic model is a pendulum made up of four fused armadillos, containing 64,275 tets and made of the same material as the previous example (Figure 4). In addition to gravity, a second body force is applied at timesteps 70-80 to induce a swinging motion. The online integrator automatically discovered a rank 15 subspace containing 67 key tets, and skipped 901 of 1500 (60.07%) timesteps. The basis has 44 snapshots added, but was downdated twice. The full integrator averaged 20.35 seconds per timestep, and the reduced integrator averaged 583 ms. A full simulation completed in 26,708 seconds, and the online integrator completed in 12,715 seconds, giving a $\mathbf{2.1}\times$ **speedup**.

The quasistatic example is a head model composed of 20,256 tets, and is composed of an Arruda-Boyce material (see e.g. [An et al. 2008]) with settings $\mu = 5000$, $N = 5000$, and $k = 100000$ (Figure 1). Skeletal subspace deformation (SSD) [Magnenat-Thalmann et al. 1988] has been used to introduce a neck joint, which then drives an EigenSkin [Kry et al. 2002] FEM simulation, i.e., the $\mathbf{Uq}$ displacement is defined in the "dress pose" of the character and is mapped through the SSD transform. For $\mathsf{T}_{quasi} = 1$, the online integrator automatically discovered a rank 12 subspace containing 143 key tets, and skipped 927 of 1000 (92.70%) timesteps. The full integrator averaged 210.00 seconds per timestep, while the reduced integrator averaged 905 ms. For $\mathsf{T}_{quasi} = 1.5$ a rank 11 subspace containing 121 key tets, was discovered, and 987 of 1000 (98.70%) timesteps skipped. The full integrator averaged 213.46 seconds per timestep, while the reduced integrator averaged 238 ms. The full simulation completed in 174,521 seconds, and the online integrator completed in 16,174 seconds for $\mathsf{T}_{quasi} = 1$ and 3,101 seconds for $\mathsf{T}_{quasi} = 1.5$, giving speedups of $\mathbf{10.79}\times$ and $\mathbf{56.28}\times$.

We additionally computed a more complex head motion that can be seen in the video. A bee lands on the character's ear and he tries various motions to shake it off. The online integrator discovered a rank 20 subspace containing 200 key tets, and skipped 528 of 600 (88.00%) timesteps. Inside the online integrator, the full integrator averaged 196.60 seconds per timestep, while the reduce integrator

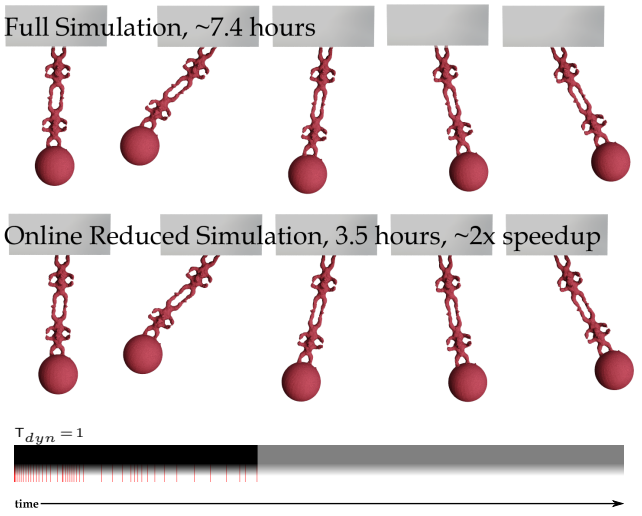


**Figure 4:** *Frames from a 1500 step pendulum simulation. The online integrator is* $\mathbf{2.1}\times$ *faster than the full simulation. Along the bottom is a visualization of the integrator steps, with the same coloring as Fig. 1. The default setting of* $\mathsf{T}_{dyn} = 1$ *skips only 60.07% of the steps. Increasing* $\mathsf{T}_{dyn}$ *would have not effect, as the skipped steps extend through the end of the simulation.*

averaged 1.42 seconds. A full simulation completed in 124,544 seconds, while our online integrator completed in 14,906 seconds, giving a $\mathbf{8.35}\times$ **speedup**.

The beam simulation used the parameter setting $\tau_{error} = 10^{-2}$, while the pendulum and quasistatic examples used the setting $\tau_{error} = 5e^{-3}$. The dynamics simulations used $\tau_{discard} = 10^{-2}$, while the quasistatic simulation used a smaller $\tau_{discard} = 10^{-3}$ to avoid excessive popping. All three examples used $r_{max} = 20$, and both dynamics examples used a timestep size of $\delta t = 10^{-2}$. All the meshes were initially normalized to a unit cube, so displacement error is always in terms of unit length.

A slight pop is visible at the beginning of the quasistatic example because the online integrator skips a step before obtaining a more robust value for $\tau_{upper}$. Larger values for $\mathsf{T}_{quasi}$ accelerate the simulation further, but the lower accuracy results in more popping. The overall global motion is still preserved, so the online integrator still serves as an effective previewing tool in this case. If speed and motion smoothness is more important than accuracy, one could also blend between the reduced and full solutions similar to KPSA [Meyer and Anderson 2007] to minimize the visual impact of the popping.

## 6 Discussion

**Newton Convergence:** All of our Newton solves were run until 4 digits of accuracy were obtained. When estimating error, it is imperative that the full solution have a known accuracy. Otherwise, it is impossible to determine if the disparity between the full and reduced solutions should be attributed to subspace error or residual error in the full solution. While we did experience problems with Newton non-convergence, these were mostly fixed by modifying the linearization point for $\hat{\mathbf{F}}$ during element inversion [Irving et al. 2004]. With appropriate inversion values, the number of Newton iterations rarely exceeded 20.

**Error compared to full rank:** The RMS error of the dynamic and quasistatic simulations compared to a full-rank simulations are

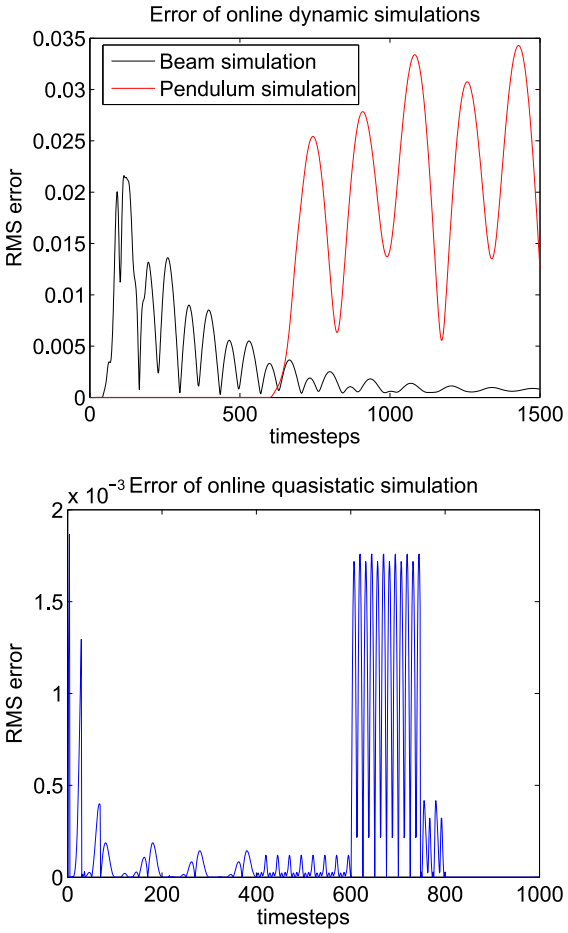


**Figure 5:** *RMS error of online simulations compared to "full rank" simulation. Error introduced into the dynamic simulation persists into later timesteps, while the quasistatic simulation is not history-dependent, so error can appear abruptly and then vanish.*

shown in Fig. 5. As described in section 4.2.2, when the dynamic online integrator introduces error into the simulation, the error persists into all later timesteps. While the error appears to decrease steadily in the beam simulation, this is because the beam is coming to rest, not because the error has necessarily been corrected. The quasistatic case shows more discontinuous behavior, because the error only depends on how far outside the basis the true equilibrium pose is, and does not accumulate smoothly over time.

**Subspace behavior under refinement:** The main attraction of the online integrator is that it can decouple the complexity of the motion from the complexity of the mesh. In order to explore the invariance of this property, we performed the simulations from Figs. 1 and Figs. 3 under different spatial and temporal resolutions (Tables 2 and 1). Higher spatial resolution was achieved by increasing the resolution of the BCC grid in an Isosurface Stuffing [Labelle and Shewchuk 2007] implementation. While the percentage of skipped steps remains highly stable for the quasistatic simulation, we found the behavior to be moderately stable to highly erratic for the dynamic simulation. Under spatial refinement, the highest resolution mesh waited the longest before skipping its first step, but was also able to pick up an additional snapshot that enabled it to skip more steps overall. Results were far more erratic under temporal refinement, where the number of skipped steps oscillated between many and none. Small tweaks restored some of the lost skipped steps. For

| BCC Res. | # tets | # vertices | $\delta t$ | $\tau_{error}$ | final rank | % skipped |
|---|---|---|---|---|---|---|
| $32^3$ | 640 | 291 | 0.01 | 0.01 | 12 | 23.5% |
| $64^3$ | 1104 | 491 | 0.01 | 0.01 | 12 | 22.8% |
| $128^3$ | 4949 | 16824 | 0.01 | 0.01 | 13 | 53.5% |
| $32^3$ | 640 | 291 | $5e^{-3}$ | $5e^{-3}$ | 15 | 0% |
| $32^3$ | 640 | 291 | $2.5e^{-3}$ | $2.5e^{-3}$ | 18 | 76.8% |
| $32^3$ | 640 | 291 | $1.25e^{-3}$ | $1.25e^{-3}$ | $12^*$ | 0% |

**Table 1:** *The dynamic beam simulation (Fig. 3) under spatial and temporal refinement.* $\mathsf{T}_{dyn} = 1$ *in all cases. Note that the rank denoted (*) was downdated once; the total number of snapshots added was 22.*

| BCC Res. | # tets | # vertices | $\delta t$ | $\tau_{error}$ | basis adds | final rank | % skipped |
|---|---|---|---|---|---|---|---|
| $32^3$ | 20256 | 4472 | 0.01 | $1e^{-4}$ | 44 | 9 | 87.7% |
| $64^3$ | 165941 | 38373 | 0.01 | $1e^{-4}$ | 30 | 20 | 87.9% |
| $32^3$ | 20256 | 4472 | $5e^{-3}$ | $1e^{-4}$ | 34 | 9 | 81.7% |
| $32^3$ | 20256 | 4472 | $2.5e^{-3}$ | $1e^{-4}$ | 37 | 7 | 84.8% |
| $32^3$ | 20256 | 4472 | $1.25e^{-3}$ | $1e^{-4}$ | 44 | 9 | 85.9% |

**Table 2:** *The quasistatic head simulation (Fig. 1) under spatial and temporal refinement.* $\mathsf{T}_{quasi} = 1$ *in all cases. Unlike the dynamic simulations, the percentage of skips is stable under refinement.*

$\delta t = 5e^{-3}$, when $\tau_{error}$ was increased from $5e^{-3}$ to $7e^{-3}$, 17.19% of the steps were skipped instead of 0%. The need for this tweaking is unfortunate, and we speculate that it occurs because smaller timesteps allow new transient waves to enter the motion, altering the subspace necessary to achieve a given $\tau_{error}$. This seems to be supported by the increased number of added snapshots as the timestep is refined, but the problem needs further study.

**Failure Cases:** We found the three most important parameters in the simulation to be $\tau_{error}$, $\tau_{discard}$ and $r_{max}$. An incorrect setting for any of these parameters would either corrupt the motion or increase the overall running time of the simulation. If $\tau_{error}$ was set too large, the integrator would take subspace steps using a poor basis. In the quasistatic case this caused extreme pops like those described at the end of Sec. 5. In the dynamics case, this dissipated away almost all of the motion, and in the worst cases returned solutions there were so poor that they caused the Newton solves in later steps to diverge. If $\tau_{error}$ was set too small, no steps were ever skipped. If $\tau_{discard}$ was set too large, no snapshots were added to the basis and no steps were skipped. If $\tau_{discard}$ was set too small, the basis became polluted with unimportant snapshots, and the subspace spanned by the basis never grew large enough to skip a significant number of steps. Similar behavior was observed if $r_{max}$ was set too small. If $r_{max}$ was too large ($> 20$), we found that the training time for the cubature scheme (line 10 of Algorithm 1) became prohibitive, and in the worst case even took longer than a full-rank simulation step. In all of our examples, the values for these parameters were arrived it via experimentation, and modifying them by an order of magnitude in either direction produced a failure case. An automatic method of determining optimal values for these parameters for an arbitrary motion remains future work.

## 7 Future Work

Using on-the-fly model reduction opens up a new direction for accelerating deformable simulations by skipping steps. As with most new approaches, there are limitations and directions for future work.

A promising area for future development is free-flight dynamics and contact handling. We demonstrated results for models with fixed vertices, however similar to exploiting simplified small-deformation or modal models in moving frames for dynamics [Terzopoulos and Witkin 1988] or collisions [James and Pai 2004], online model reduction should be able to exploit the redundancy of motion in moving frames, and transition from rigid to reduced-deformable to general models seamlessly [Ogot et al. 1996]. Collisions and contact pose special challenges depending on the resulting motion complexity, and contact resolution methods employed. However, there is evidence that reduced-order models might be exploited in frictional contact resolution [Kaufman et al. 2008].

Building reduced-order models on-the-fly can help skip steps for low-rank and redundant motion, however it is not a panacea for deformable simulation: complex motions with higher rates of subspace expansion are less likely to be accelerated by such methods. For example, the speedup observed in the pendulum example is smaller than the two other examples because the motion is inherently of higher rank. In general, exploiting sparse $\mathbf{U}$ basis matrices for large models is appealing. In some cases, it may be possible to cluster the mesh into several lower rank submeshes at runtime and integrate them in parallel, and extending reduced-order domain decomposition methods [Huang et al. 2006] to the online model reduction setting is a direction for future work.

Finally, it remains to be seen if a similar online algorithm can be designed for reduced-order fluid dynamics [Treuille et al. 2006; Wicke et al. 2009]. Different error estimators had to be designed for the dynamic and quasistatic cases here, so it is likely that a different estimator will be needed for the case of fluids.

**Acknowledgements** The authors would like to extend a special thanks to Scott Wells (`http://www.scottwells3d.com`) for generously providing access to his "Les Avocats" model in Figure 1. We would like to thank the anonymous reviewers for helpful feedback. This work was supported in part by the National Science Foundation (CAREER-0430528, EMT-CompBio-0621999), the National Institutes of Health (NIBIB/NIH R01EB006615), the Alfred P. Sloan Foundation, and generous donations by Pixar, Intel, and Autodesk. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing Cubature for Efficient Integration of Subspace Deformations. *ACM Trans. on Graphics 27*, 5 (Dec.), 165.

BALAY, S., BUSCHELMAN, K., GROPP, W. D., KAUSHIK, D., KNEPLEY, M. G., MCINNES, L. C., SMITH, B. F., AND ZHANG, H., 2001. PETSc Web page. http://www.mcs.anl.gov/petsc.

BARAFF, D., AND WITKIN, A. P. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 43–54.

BARBIČ, J., AND JAMES, D. L. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. on Graphics 24*, 3 (Aug.), 982–990.

BARBIČ, J., AND JAMES, D. L. 2007. Time-critical distributed contact for 6-dof haptic rendering of adaptively sampled reduced deformable models. In *ACM SIGGRAPH Symposium on Computer Animation, San Diego, CA*.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics*, 594–603.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A Multiresolution Framework for Dynamic Deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, 41–48.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. In *Proc. of ACM SIGGRAPH 2001*, 31–36.

GIBSON, S. F., AND MIRTICH, B. 1997. A Survey of Deformable Models in Computer Graphics. Tech. Rep. TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, November.

GOLUB, G., AND VAN LOAN, C. 1996. *Matrix Computations*, third ed. The Johns Hopkins University Press, Baltimore.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Trans. on Graphics 21*, 3 (July), 281–290.

HOMESCU, C., PETZOLD, L., AND SERBAN, R. 2006. Error estimation for reduced-order models of dynamical systems. *SIAM Journal of Numerical Analysis 43*, 4, 1693–1714.

HUANG, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. An efficient large deformation method using domain decomposition. *Computers and Graphics 30*, 927–935.

IDELSOHN, S., AND CARDONA, A. 1985. A reduction method for nonlinear structural dynamic analysis. *Computer Methods in Applied Mechanics and Engineering 49*, 253–279.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *ACM SIGGRAPH Symposium on Computer Animation*, 131–140.

JAMES, D. L., AND PAI, D. K. 2004. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics 23*, 3 (Aug.), 393–398.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Transactions on Graphics 24*, 3 (Aug.), 399–407.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. *ACM Trans. on Graphics 27*, 5 (Dec.), 164:1–164:11.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *ACM SIGGRAPH Symposium on Computer Animation*, 153–160.

KRYSL, P., LALL, S., AND MARSDEN, J. E. 2001. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering 51*, 479–504.

LABELLE, F., AND SHEWCHUK, J. R. 2007. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics 26*, 3 (Aug.), 57.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation. In *Proceedings of ACM SIGGRAPH 2000*, 165–172.

MAGNENAT-THALMANN, N., LAPERRIÈRE, R., AND THALMANN, D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface*, 26–33.

MEYER, M., AND ANDERSON, J. 2007. Key Point Subspace Acceleration and Soft Caching. *ACM Transactions on Graphics 26*, 3 (July), 74.

MEYER, M., AND MATTHIES, H. G. 2003. Efficient model reduction in non-linear dynamics using the Karhunen-Lóeve expansion and dual-weighted-residual methods. *Computational Mechanics 31*, 179–191.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. on Graphics 23*, 3 (Aug.), 385–392.

NEALEN, A., MULLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Eurographics: State of the Art Report*.

O'BRIEN, J., AND HODGINS, J. 1999. Graphical modeling and animation of brittle fracture. *ACM Trans. on Graphics*, 137–146.

OGOT, M., LIANG, Y., AND CUITINO, A. 1996. Hybrid simulation strategy for multiple planar collisions with changing topologies and local deformation. *Finite Elements in Analysis & Design 23*, 2-4, 225–239.

RYCKELYNCK, D., HERMANNS, L., CHINESTA, F., AND ALARCÓN, E. 2005. An efficient a priori model reduction for boundary element models. *Engineering Analysis with Boundary Elements 29*, 796–801.

RYCKELYNCK, D. 2005. A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics 202*, 1, 346 – 366.

RYCKELYNCK, D. 2009. Hyper-reduction of mechanical models involving internal variables. *International Journal for Numerical Methods in Engineering 77*, 75–89.

SHABANA, A. A. 1990. *Theory of Vibration, Volume II: Discrete and Continuous Systems*. Springer–Verlag, New York, NY.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. In *ACM SIGGRAPH Symposium on Computer Animation*, 181–190.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Deformable models. *The Visual Computer 4*, 6 (Dec.), 306–331.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically Based Models with Rigid and Deformable Components. *IEEE Computer Graphics & Applications 8*, 6 (Nov.), 41–51.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically Deformable Models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 205–214.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics 25*, 3 (July), 826–834.

UTKU, S., CLEMENTE, J., AND SALAMA, M. 1985. Errors in reduction methods. *Computers and Structures 21*, 6, 1153–1157.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-Weight Enveloping: Least-Squares Approximation Techniques for Skin Animation. In *ACM SIGGRAPH Symposium on Computer Animation*, 129–138.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Trans. on Graphics 28*, 3 (Aug.), 39.