SIGGRAPH 2012 Course Notes
FEM Simulation of 3D Deformable Solids: A practitioner's guide to theory, discretization and model reduction.
Part 2: Model Reduction (version: August 4, 2012)

Jernej Barbič

Course notes URL: `http://www.femdefo.org`

1 Introduction to model reduction

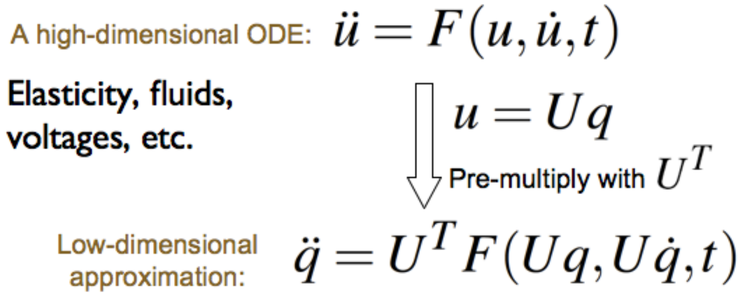

A high-dimensional ODE: $\ddot{u} = F(u, \dot{u}, t)$
Elasticity, fluids, voltages, etc.
$u = Uq$
Pre-multiply with U^T
Low-dimensional approximation: $\ddot{q} = U^T F(Uq, U\dot{q}, t)$

Figure 1: **Model reduction overview:** a high-dimensional ordinary differential equation is approximated with a projection to a low-dimensional space.

Model reduction (also called dimensional model reduction, or model order reduction (MOR)) is a technique to simplify the simulation of dynamical systems described by differential equations. The idea is to project the original, high-dimensional, state-space onto a properly chosen low-dimensional subspace to arrive at a (much) smaller system having properties similar to the original system (see Figure 1). Complex systems can thus be approximated by simpler systems involving fewer equations and unknown variables, which can be solved much more quickly than the original problem. Such projection-based model reduction appears in literature under the names of *Principal Orthogonal Directions (POD) Method*, or *Subspace Integration Method*, and it has a long history in the engineering and applied mathematics literature [29]. See [27] and [33] for good overviews of model reduction applied to linear and nonlinear problems, respectively.

Model reduction has been used extensively in the fields of control theory, electrical circuit simulation, computational electromagnetics and microelectromechanical systems [28]. Most model reduction techniques in these fields, however, aim at linear systems, and linear time-invariant systems in particular, e.g., small perturbations of voltages in some complex nonlinear circuit. Another common characteristic of these applications is that both the input and output are low-dimensional, i.e., one may want to study how the voltage level at some circuit location depends on the input voltage at another location, in a complex nonlinear circuit. In *computer graphics*, however, one is often interested in nonlinear systems (e.g., large deformations of objects) that exhibit interesting, very visible, dynamics. The output in computer graphics is usually high-dimensional, e.g., the deformation of an entire 3D solid object, or fluid velocities sampled on a high-resolution grid. For these reasons, many conventional reduction techniques do not immediately apply to computer graphics problems.

1.1 Survey of POD-based model reduction in computer graphics

The initial model reduction applications to deformable object simulation in computer graphics investigated *linear* FEM deformable objects [32, 18, 14]. These models are very fast, but are (due to linear Cauchy strain) accurate only for small deformations and produce visible artifacts under large deformations. In order to avoid such artifacts, it is necessary to apply reduction to *nonlinear* elasticity. For *real-time* geometrically nonlinear deformable objects (quadratic Green-Lagrange strain), such an approach was presented by Barbič and James [6, 4], who also gave an automatic approach to select a quality low-dimensional basis, using *modal derivatives*. An and colleagues [2] demonstrated how to efficiently support arbitrary nonlinear material models. Model reduction has also been used for fast sound simulation [17, 10] and to simulate frictional contact between deformable objects [21]. For deformable FEM offline simulations, Kim and James [23] applied *online* model reduction to adaptively replace expensive full simulation steps with reduced steps, which made it possible to *throttle* the simulation costs at run-time. Treuille and colleagues applied model reduction to fluid simulation in computer graphics [39]. Wicke and colleagues [41] improved Treuille's fluid method to support reduced fluid simulations on several (inter-connected) domains with specialized basis functions on each domain (domain decomposition for fluids). Recently, Barbič and Zhao [8] demonstrated a domain decomposition method for open-loop solid deformable models, by employing gradients of polar decomposition rotation matrices, whereas Kim and James [24] tackled a similar problem using inter-domain spring forces.

2 Linear modal analysis

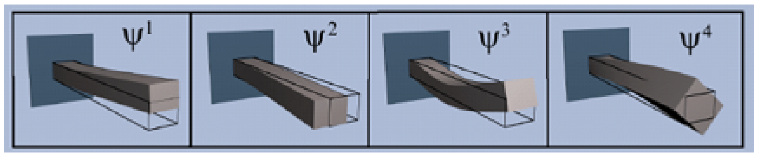


Figure 2: **Linear modes** for a cantilever beam.

Although elastic objects can in principle deform arbitrarily, they tend to have a bias in deforming into certain characteristic, low-energy shapes. Most of us will remember the high-school physics example of a string stretched between two walls (or, say, a violin string), where one studies the natural frequencies ω_i of the string, together with their associated shapes, typically of the form $\sin(\omega_i x)$ and $\cos(\omega_i x)$. The same intuition carries over to arbitrary three-dimensional elastic objects deforming by a small amount around their rest configuration, whether it be a metal wire, a thin shell (e.g., cloth on a character), or a solid 3D tet mesh model of a skyscraper. The low-frequency *modes* are the deformations, which, for a given amount of displaced mass (or volume), subject to specific boundary conditions such as fixed vertices, increase the elastic strain energy of the object by the least amount. In other words, they are the shapes with the *least* resistance to deformation. How are the modes and frequencies computed? One has to first form the system mass and stiffness matrices $M \in \mathbb{R}^{3n\times 3n}$ and $K \in \mathbb{R}^{3n\times 3n}$, where n is the number of mesh vertices. The specific approach to compute M and K depends on each particular mechanical system. For example, the tet skyscraper may be modeled using 3D FEM elasticity, whereas for a cloth model one may compute M by lumping the mass at the vertices, and set K to the gradient of the internal cloth forces (in the rest configuration), computed, say, using the Baraff-Witkin cloth model [3]. Once M and K are known, one has to prescribe boundary conditions, i.e., specify how the object is constrained. The modes and frequencies greatly depend on this choice. It is possible to set no boundary conditions, in which case one obtains *free-flying modes*. In order to compute the modes, one forms matrices $\overline{M}$ and $\overline{K}$ where the rows and columns corresponding to the fixed degrees of freedom have been removed from M and K. Then, one solves the generalized eigenvalue problem

$$\overline{K}x = \lambda \overline{M}x. \tag{1}$$

Matrices $\overline{M}$ and $\overline{K}$ are typically large and sparse. One can solve the eigenvalue problem, say, using the Arnoldi iteration implemented by the ARPACK eigensolver [26]. This solver is free, and has performed very well in various model reduction computer graphics projects by Jernej Barbič and other researchers in the field. Because $\overline{M}$ and $\overline{K}$ are symmetric positive-definite (in typical applications, e.g., deformable object in the rest configuration), the eigenvalues are real and non-negative. One seeks the smallest eigenvalues λ_i and their associated eigenvectors ψ_i, $i = 1, 2, \ldots, k$, where k is the number of modes to be retained. For objects with no constrained vertices (free-flying objects), the first six eigenvalues are zero and the modes correspond to rigid translations and infinitesimal rotations; these modes are typically discarded. The eigenvalues are squares of the natural frequencies of vibration, $\lambda_i = \omega_i^2$, and the eigenvectors ψ_i are the modes. It should be noted that one typically inserts zeros into ψ_i at locations of fixed degrees of freedom, so that the resulting vector is of length $3n$. In order to check that the eigensolver was successful, it is common to visualize the individual modes, by animating them as $\psi_i \sin(\omega_i t)$, where t is time. The different modal vectors are typically assembled into a *linear modal basis* matrix $U = [\psi_1, \ldots, \psi_k] \in \mathbb{R}^{3n \times k}$.

2.1 Small deformation simulation using linear modal analysis

Small deformations $u \in \mathbb{R}^{3n}$ (n is the number of mesh vertices) follow the equation

$$M\ddot{u} + D\dot{u} + Ku = f, \tag{2}$$

where $M, D, K \in \mathbb{R}^{3n \times 3n}$ are the mass, damping and stiffness matrices, respectively, and $f \in \mathbb{R}^{3n}$ are the external forces [35]. Equation 2 is a linear, high-dimensional ordinary differential equation, obtained by applying the Finite Element Method (FEM) to the linearized partial differential equations of elasticity. It is most commonly applied to 3D solids, but can also model shells and strands. It is only accurate under small deformations; very visible artifacts appear under large deformations. Equation 2 models the object at full resolution (no reduction), which means that it incorporates transient effects such as (localized) waves traveling across the object. It is often employed, for example, to perform earthquake simulation, typically using supercomputers on large meshes involving millions of degrees of freedom [1]. For real-time applications, the computational costs of timestepping Equation 2 may be prohibitive. Instead, one can perform model reduction, by approximating the deformation vector u as $u = Uz$, where $z \in \mathbb{R}^k$ is a vector of *modal amplitudes* (typically $k \ll 3n$). If we choose damping to be a linear combination of M and K, $D = \alpha M + \beta K$, for some scalars $\alpha, \beta > 0$ (this is called *Rayleigh damping*), and pre-multiply Equation 2 by U^T, Equation 2 projects to *k independent* one-dimensional ordinary differential equations

$$\ddot{z}_i + (\alpha + \beta\lambda_i)\dot{z}_i + \lambda_i z_i = \psi_i^T f, \tag{3}$$

for $i = 1, \ldots, k$. Here, we have used the fact that the modes are generalized eigenvectors $K\psi_i = \lambda_i M\psi_i$, and are therefore mass-orthonormal, $(M\psi_i)^T\psi_j = 0$ when $i \neq j$, and $(M\psi_i)^T\psi_i = 1$ for all i. The one-dimensional equations given in (3) can be timestepped independently. This can be done very efficiently (see, e.g., [18]). The full deformation can be reconstructed by multiplying $u = Uz$. This multiplication is fast when k is small (a few hundred modes). It can also be performed very efficiently in graphics hardware [18]. It can be shown that as $k \to 3n$, this approximation converges to the solution of Equation 2. This property is very useful, as it makes it possible to trade computation accuracy for speed.

2.2 Application to sound simulation

Sound originates from mechanical vibration of objects. These mechanical vibrations excite the surrounding medium (typically air or water), and the pressure waves then propagate to the listener location. The object mechanical vibrations are usually modeled using FEM and the equations of elasticity, whereas the pressure propagation is usually modeled by the wave equation. There are varying degrees of approximation that can be applied to each of these two tasks. Because deformation amplitudes in sound applications are small, linear elasticity is often employed for their simulation [31, 17]. However, richer, nonlinear sound can be produced using nonlinear simulation [10]. Linear simulation is straightforward and follows the material from Section 2.1. For each object that is to produce sound, one first has to set the fixed vertices (if any), and extract the modes. Next, one runs any physical simulation (typically rigid body simulation), producing contact forces. These forces are then used as external forces f for the modal oscillators in Equation 3, producing modal excitations $z_i(t)$. It remains to be described how $z_i(t)$ are used to generate the sound signal $s(t)$. A very common approach is to assign some meaningful weights w_i to each mode, and compute sound as

$$s(t) = \sum_{i=1}^{k} w_i z_i(t), \tag{4}$$

where k is the number of modes. The weights w_i can be set to a constant, $w_i = 1$ [31], or they can be made non-constant to model the fact that different modes radiate with different intensities. Alternatively, weights can be made to depend on the listener location x, $w_i = w_i(x,t)$, by solving the spatial part of the wave equation (Helmholtz equation) [17, 10]. Such spatially-dependent weights can model diffraction of sound around the scene geometry.

3 Model reduction of nonlinear deformations

Up to this point, we have considered model reduction of linear systems (Equation 2). Linear systems have an important limitation: they produce very visible artifacts under large deformations (see Figure 3). These artifacts can be removed by applying model reduction to the nonlinear equations of a deformable object:

$$M\ddot{u} + D\dot{u} + f_{\text{int}}(u) = f. \tag{5}$$

As detailed in the first part of the course, the nonlinearity in the internal forces $f_{\text{int}}(u)$ arises due to large-deformation strain (*geometric nonlinearity*), and due to nonlinearities in the strain-stress relationship. How to apply model reduction to Equation 5? We proceed in the same way as with linear systems; we assume the availability of a basis $U \in \mathbb{R}^{3n \times r}$ (r is basis size) and approximate $u = Uz$, where $z \in \mathbb{R}^r$ is the vector of *reduced coordinates*. After projection by U^T, we obtain

$$\ddot{z} + U^T D U \dot{z} + U^T f_{\text{int}}(Uz) = U^T f. \tag{6}$$

This equation determines the dynamics of the reduced coordinates $z = z(t) \in \mathbb{R}^r$, and therefore also the dynamics of $u(t) = Uz(t)$. Equation 5 is similar to Equation 3, except that it is nonlinear and the components of z are **not** decoupled. At this point, two questions emerge: (1) How can we timestep Equation 6? (2) How do we choose the basis U?

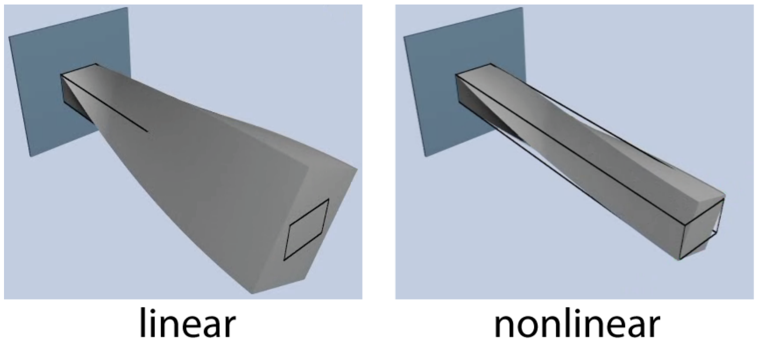


Figure 3: **Model reduction applied to a linear and nonlinear system.**

3.1 Timestepping the reduced nonlinear equations of motion

In order to timestep Equation 5, one needs to evaluate the *reduced internal forces*, $\tilde{f}_{\text{int}} = U^T f_{\text{int}}(Uz)$ for arbitrary configurations $z \in \mathbb{R}^r$. Furthermore, as equations of elasticity are typically stiff, implicit integration is required, necessitating a further derivative of $\tilde{f}_{\text{int}}$, the *reduced tangent stiffness matrix*

$$\tilde{K}(z) = \frac{d\tilde{f}_{\text{int}}}{dz} = U^T K(Uz) U \in \mathbb{R}^{r \times r}. \tag{7}$$

Here $K(u) = df_{\text{int}}/du$ is the (unreduced) tangent stiffness matrix in configuration u. Note that in general, the term $\tilde{f}_{\text{int}}$ cannot be algebraically simplified; its evaluation must proceed by first forming Uz, then evaluating $f_{\text{int}}(Uz)$ and finally forming a projection by pre-multiplying with U^T. Evaluation of $\tilde{K}(z)$ is even more complex. Once $\tilde{f}_{\text{int}}(z)$ and $\tilde{K}(z)$ are known, one can use any implicit integrator to timestep the system (see [4] for details). The key important fact is that this integrator will need to solve a dense $r \times r$ linear system as opposed to a sparse $3n \times 3n$ system as is the case with implicit integration of unreduced systems. Since $r \ll 3n$, this usually leads to significant computational savings.

How to evaluate $\tilde{f}_{\text{int}}(z)$ and $\tilde{K}(z)$ in practice? If the simulation is geometrically nonlinear, but materially *linear*, then it can be shown [4] that each component of $f_{\text{int}}(u)$ is a *cubic polynomial* in the components of u [4]. Consequently, $\tilde{f}_{\text{int}}$ are cubic polynomials in the components of z. Note that this is a manifestation of a more general principle: for any polynomial function $G(u)$, its projection $U^T G(Uz)$ will be a polynomial in z, of the same degree. Treuille and colleagues, for example, exploited this fact with quadratic advection forces for reduced fluids [39]. For geometrically nonlinear materials, one can precompute the coefficients of the cubic polynomials. As there are r components of the reduced force, each of which is a cubic polynomial in r variables, the necessary storage is $O(r^4)$. For moderate values of r ($r < 30$), this storage is manageable (under one megabyte; details are in [4]). Because the reduced stiffness matrix $\tilde{K}(z)$ is the gradient of $\tilde{f}_{\text{int}}$ with respect to z, the reduced stiffness matrix is a quadratic function in z with coefficients directly related to those of the reduced internal forces. For exact evaluation of internal forces and tangent stiffness matrices, all polynomial terms must be "touched" exactly once. Therefore, the cost of evaluation of reduced internal forces and tangent stiffness matrices is $O(r^4)$, whereas the cost of implicit integration is $O(r^3)$.

For general materials, An and colleagues [2] have designed a fast approximation scheme which can decrease the reduced internal force and stiffness matrix computation time to $O(r^2)$ and $O(r^3)$, respectively. For simulations that use implicit integration, the runtime complexity is therefore $O(r^3)$. The method works by observing that the elastic strain energy $E(z)$ and internal forces $\tilde{f}_{\text{int}}(z)$, for reduced coordinates z, are obtained by integration of the energy density $\Psi(X, z)$ and its gradient over the entire mesh:

$$E(z) = \int_\Omega \Psi(X, z) dV, \tag{8}$$

$$\tilde{f}_{\text{int}}(z) = \int_\Omega \frac{\partial \Psi(X, z)}{\partial z} dV. \tag{9}$$

As opposed to evaluating $\tilde{f}_{\text{int}}$ using the exact formula $\tilde{f}_{\text{int}} = U^T f_{\text{int}}(Uz)$, An and colleagues approximate the integral in (9) using numerical quadrature. In order to do so, they determine positions $X_i \in \Omega$, and weights $w_i \in \mathbb{R}$, such that

$$\tilde{f}_{\text{int}}(z) = \int_\Omega \frac{\partial \Psi(X, z)}{\partial z} dV \approx \sum_{i=1}^{T} w_i\, g(X_i, z), \tag{10}$$

$$\tilde{K}(z) \approx \sum_{i=1}^{T} w_i \frac{\partial g(X_i, z)}{\partial z}, \tag{11}$$

where $g(X, z) = \partial \Psi(X, z)/\partial z$. At runtime, given a value z, one then only has to evaluate $g(X_i, z)$ and $\partial g(X_i, z)/\partial z$, for $i = 1, \ldots, T$ and sum the terms together. The number of quadrature points T is usually set to $T = r$. Positions and weights are obtained using a training process. Given a set of representative "training" reduced coordinates $z^{(1)}, \ldots, z^{(N)}$, the method computes positions and weights that best approximate the reduced force $\tilde{f}_{\text{int}}$ for these training datapoints. To avoid overfitting and to keep the stiffness matrix symmetric positive-definite, the weights w_i are chosen to be non-negative, using nonnegative least squares (NNLS) [25]. The positions X_i are determined using a greedy approach, designed to minimize the NNLS error residual (details in [2]).

3.2 Choice of basis

The matrix U is a time-invariant matrix specifying a basis of some r-dimensional ($r \ll 3n$) linear subspace of $\mathbb{R}^{3n}$. The basis is assumed to be mass-orthonormal, i.e., $U^T M U = I$. If this is not the case, one can easily convert U to such a basis using a mass-weighted Gramm-Schmidt process. For each fixed $r > 1$, there is an infinite number of possible choices for the linear subspace and for its basis. Good subspaces are low-dimensional spaces which well-approximate the space of typical nonlinear deformations. The choice of subspace depends on geometry, boundary conditions and material properties. Selection of a good subspace is a non-trivial problem. We now present two choices: basis from simulation data ("POD basis"), and basis from modal derivatives. The former requires pre-simulation (using a general deformable solver), whereas the latter can create a basis automatically without pre-simulation.

3.2.1 Basis from simulation data ("POD basis")

In model reduction literature, a very common approach to create a basis for nonlinear systems is to obtain some "snapshots" of the system, $u_1, u_2, \ldots, u_N$, and then use statistical techniques to extract a representative low-dimensional space. The snapshots can be obtained by running a full (unreduced) simulation, or using measurements of a real system. Given the snapshots, one obtains the subspace U by performing singular value decomposition (SVD) on $A = [u_1, u_2, \ldots, u_N]$,

$$A = U \Sigma V^T, \tag{12}$$

and retains the columns of U corresponding to the largest r singular values. In practice, it is advantageous to apply SVD not with respect to the standard inner product in $\mathbb{R}^{3n}$, but with respect to a mass-weighted inner-product $<Mx, y>$ (*mass-PCA*; see [4] for details).

It is challenging to measure transient volumetric deformation fields with high accuracy [22, 9]. Therefore, large-deformation model reduction applications in computer graphics have so far relied on simulation data, or on modal derivatives, which we describe next.

3.2.2 Modal derivatives

Linear modal analysis (Section 2) provides a quality deformation basis for small deformations away from the rest pose. The linear modes, however, are not a good basis for large deformations, because they lack the deformations that automatically "activate" in a nonlinear system. For example, when a cantilever beam deflects sideways in the direction of the first linear mode, it also simultaneously compresses, in a very specific, non-uniform way. This happens automatically in a nonlinear system. A linear basis, however, lacks the proper (non-uniform) compression mode, and therefore the system projected onto the linear basis will be stiff (it "locks"). In practice, such locking manifests as a rapid loss of energy (numerical damping), and as an increase in the natural oscillation frequencies of the system, a phenomenon also observed with model reduction of electrical circuits [13]. One could attempt to resolve these issues by retaining a larger number of linear modes. Such an approach is, however, not very practical with nonlinear systems, because a very large number of modes would be needed in practice, whereas the time to solve the reduced system for implicit integration scales as $O(r^3)$.

These problems can be remedied using modal derivatives: deformations that naturally co-appear in a nonlinear system when the system is excited in the direction of linear modes. By forming a basis that consists of both linear modes and their modal derivatives, we arrive at a compact, low-dimensional basis, that can represent large deformations and that can be computed purely based on the mesh geometry and material properties; no advance knowledge of run-time forcing or pre-simulation is required. We will illustrate modal derivatives for deformable objects that are sufficiently constrained so that they do not possess rigid degrees of freedom, but modal derivatives can also be computed for unconstrained systems. Under a static load f, the system will deform into a deformation u, where u satisfies the unreduced static equation $f_{\text{int}}(u) = f$. Consider what happens if we statically load the system into the direction of

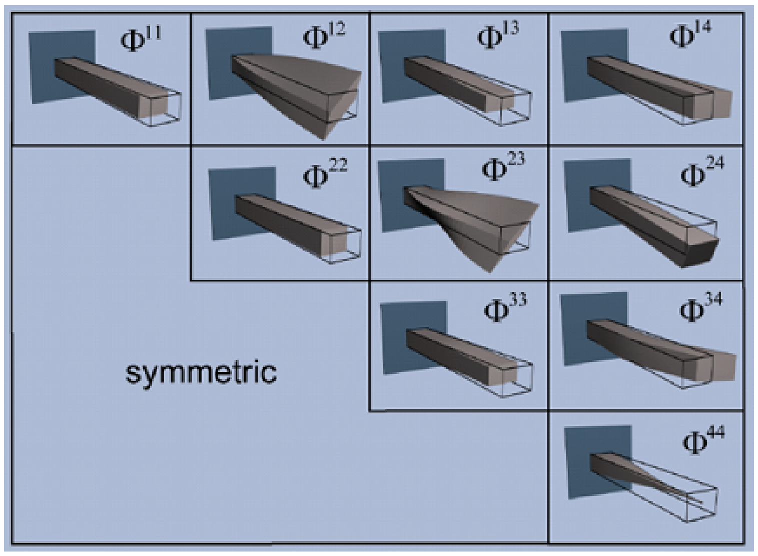


Figure 4: **Modal derivatives** for a cantilever beam.

linear modes. In particular, suppose we apply a static force load $MU_{\text{lin}}\Lambda p$, where M is the mass matrix, $U_{\text{lin}} = [\psi_1, \psi_2, \ldots, \psi_k]$ is the linear modal matrix, Λ is the diagonal matrix of squared frequencies $\text{diag}(\omega_1^2, \ldots, \omega_k^2)$, and $p \in \mathbb{R}^k$ is some parameter that controls the strength of each mode in the load. It can be easily verified that these are the force loads which, for small deformations, produce deformations within the space spanned by the linear modes. Given a p, we can solve the nonlinear equation $f_{\text{int}}(u) = MU_{\text{lin}}\Lambda p$ for u, i.e., we can define a unique function $u = u(p)$ (mapping from $\mathbb{R}^k$ to $\mathbb{R}^{3n}$, and C^∞ differentiable), such that

$$f_{\text{int}}(u(p)) = MU_{\text{lin}}\Lambda p, \tag{13}$$

for every $p \in \mathbb{R}^k$ in some sufficiently small neighborhood of the origin in $\mathbb{R}^k$. Can we compute the Taylor series expansion of u in terms of p? By differentiating Equation 13 with respect to p, one obtains

$$\frac{\partial f_{\text{int}}}{\partial u}\frac{\partial u}{\partial p} = MU_{\text{lin}}\Lambda, \tag{14}$$

which is valid for all p in some small neighborhood of the origin of $\mathbb{R}^k$. In particular, for $p = 0^k$, we get $K\frac{\partial u}{\partial p} = MU_{\text{lin}}\Lambda$. Therefore, $\frac{\partial u}{\partial p} = U_{\text{lin}}$, i.e., the first-order response of the system are the linear modes, as expected. To compute the second order derivatives of u, we differentiate Equation 14 one order further by p, which, when we set $p = 0^k$, gives us

$$K\frac{\partial^2 u}{\partial p_i \partial p_j} = -(H : \psi_j)\psi_i. \tag{15}$$

Here, H is the *Hessian stiffness tensor*, the first derivative of the tangent stiffness matrix, evaluated at $u = 0$ (see [4]). The deformation vectors

$$\Phi^{ij} = \frac{\partial^2 u}{\partial p_i \partial p_j} \tag{16}$$

are called *modal derivatives*. They are symmetric, $\Phi_{ij} = \Phi_{ji}$, and can be computed from Equation 15 by solving linear systems with a constant matrix K (stiffness matrix of the origin). Because K is constant and symmetric positive-definite, it can be pre-factored using Cholesky factorization. One can then rapidly (in parallel if desired) compute all the modal derivatives, $0 \le i \le j < k$. Note that the modal derivatives are, by definition, the second derivatives of $u = u(p)$. The second-order Taylor series expansion is therefore

$$u(p) = \sum_{i=1}^{k} \Psi^i p_i + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \Phi^{ij} p_i p_j + O(p^3). \tag{17}$$

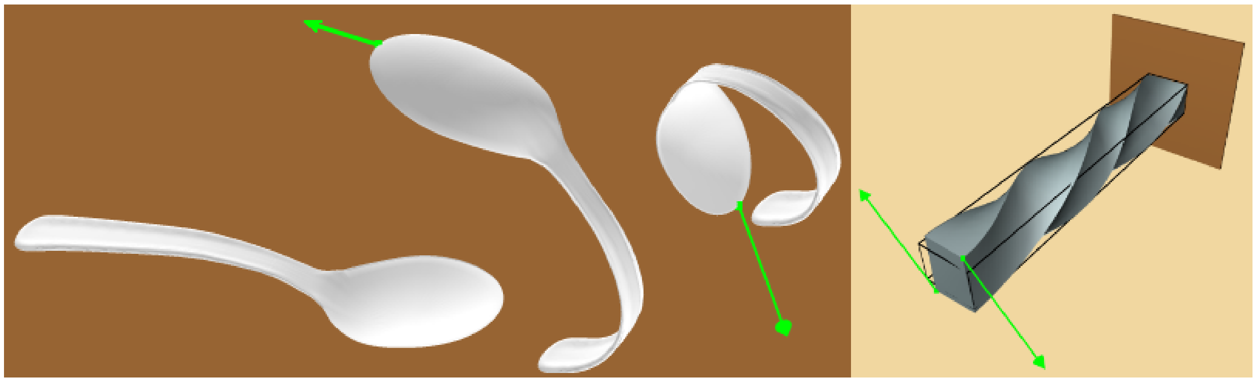


Figure 5: **Extreme shapes captured by modal derivatives:** Although modal derivative are computed about the rest pose, their deformation subspace contains sufficient nonlinear content to describe large deformations. Left: Spoon ($k = 6, r = 15$) is constrained at far end. Right: Beam ($r = 5$, twist angle=270°) is simulated in a subspace spanned by "twist" linear modes and their derivatives $\Psi^4, \Psi^9, \Phi^{44}, \Phi^{49}, \Phi^{99}$.

The modal derivatives, together with the linear modes, therefore span the natural second-order system response for large deformations around the origin.

Creating the basis U**:** Equation 17 suggests that the linear space spanned by all vectors Ψ^i and Φ^{ij} is a natural candidate for a basis (after mass-Gramm-Schmidt mass-orthonormalization). However, its dimension $k + k(k+1)/2$ may be prohibitive for real-time systems. In practice, we obtain a smaller basis by scaling the modes and derivatives according to the eigenvalues of the corresponding linear modes, and applying mass-PCA on the "dataset"

$$\left\{ \frac{\lambda_1}{\lambda_j} \Psi^j \mid j = 1, \ldots, k \right\} \cup \left\{ \frac{\lambda_1^2}{\lambda_i \lambda_j} \Phi^{ij} \mid i \leq j;\ i, j = 1, \ldots, k \right\}. \tag{18}$$

The scaling puts greater weight on dominant low-frequency modes and their derivatives, which could otherwise be masked by high-frequency modes and derivatives.

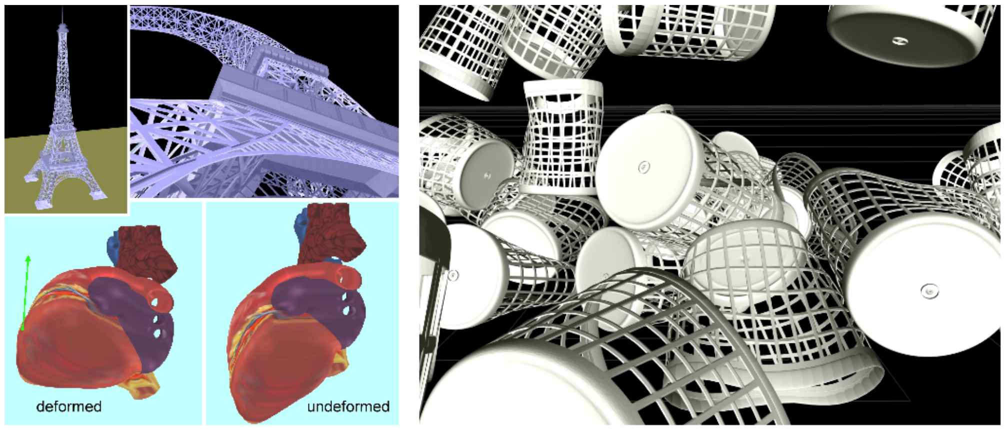


Figure 6: **Reduced simulations:** Left: Model reduction enables interactive simulations of nonlinear deformable models. Right: reduction also enables fast large-scale multibody dynamics simulations, with nonlinear deformable objects undergoing free flight motion. Collisions among the 512 baskets were resolved using BD-Trees [20].

4 Model reduction and domain decomposition

Model reduction as described in the previous section is global: it reduces the entire object using a single, global basis. Unless r is large, it is difficult to capture local detail using such a basis. Because the computation time grows at least as $O(r^3)$ (time to solve the dense $r \times r$ system [2]), large values of r (several hundreds of modes) are not practical. It is therefore natural to ask if the object can somehow be decomposed into smaller pieces, each of which is reduced separately, and the pieces are then connected into a global system. This is the idea of *domain decomposition*, a classical technique in applied mathematics and engineering. In engineering applications, however, the deformations are typically small. In computer graphics, we have to accommodate large deformations (e.g., rotations) in the interfaces joining two domains, which means that standard domain decomposition techniques cannot simply be extended to computer graphics problems.

In computer graphics, domain decomposition for deformable models has initially been applied to small domain deformations and with running times dependent on the number of domain and interface vertices. For example, a linear quasi-static application using Green's functions has been presented in [19], whereas Huang and colleagues [15] exploited redundancy in stiffness matrix inverses to combine linear FEM with domain decomposition. Recently, domain decomposition under large deformations has received significant attention in computer graphics literature. Barbič and Zhao [8] demonstrated a domain decomposition method by employing gradients of polar decomposition rotation matrices, whereas Kim and James [24] tackled a similar problem using inter-domain spring forces.

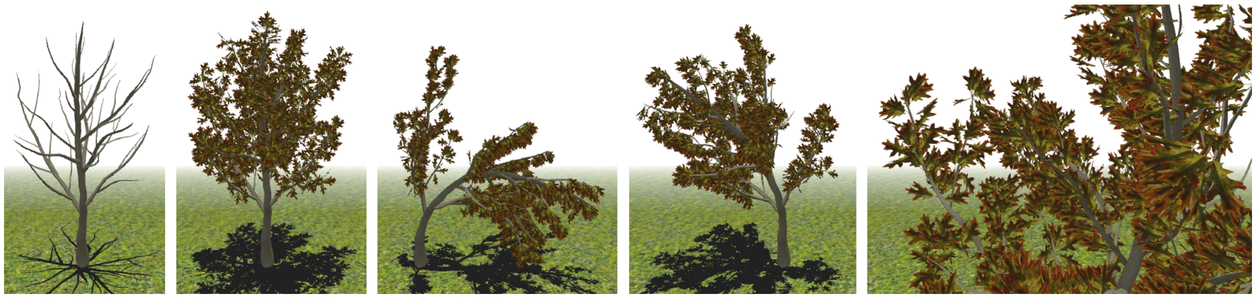


Figure 7: **Model reduction with a large number of localized degrees of freedom:** Left: nonlinear reduced simulation of an oak tree (41 branches ($r = 20$), 1394 leaves ($r = 8$), $d = 1435$ domains, $\hat{r} = 11,972$ total DOFs) running at 5 fps. Right: simulation detail.

5 Model reduction and control

Optimal control problems occur frequently in computer animation. Often, they are cast as *space-time optimization problems* involving human motion [34], fluids [40, 30] and deformations [42]. With optimal control of deformable objects, one seeks a sequence of forces (*control vectors*) $f_i \in \mathbb{R}^{3n}$, $i = 0, ..., T-1$, such that the resulting deformations (*state vectors*) $u_i \in \mathbb{R}^{3n}$, $i = 0, ..., T-1$, obtained by timestepping Equation 5 forward in time under those forces, minimize some scalar objective $E(u_0, ..., u_{T-1})$. The scalar objective typically includes terms such as magnitude of control vectors, deviation from some reference trajectory, deviation from keyframes at specific moments of time, and magnitude of deformation velocities and accelerations. The forces are sometimes expressed as $f_i = Bg_i$, where the matrix $B \in \mathbb{R}^{3n \times m}$ gives the *control basis*. The problem is said to be *underactuated* when $m < 3n$ and *fully actuated* for $m = 3n$. Underactuated problems can model objects that can propel themselves using "muscles", and are generally much more difficult to solve than fully actuated problems. Because optimal control problems compound both space and time, they have a very high dimensionality: there are $3nT$ unknowns (the control vectors f_i) in the optimal control problem. Such a huge state space leads to optimization problems that diverge, converge to local minima, or take a very long time to converge to a plausible solution.

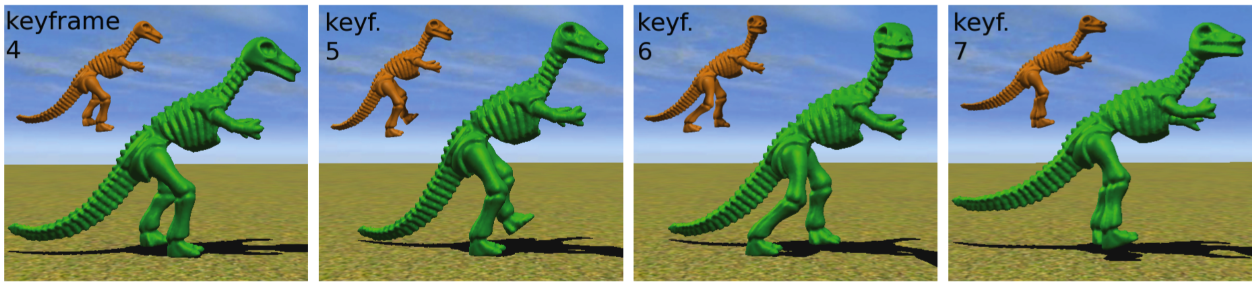


Figure 8: **Fast authoring of animations with dynamics [5]:** This soft-body dinosaur sequence consists of five walking steps, and includes dynamic deformation effects due to inertia and impact forces. Each step was generated by solving a space-time optimization problem, involving 3 user-provided keyframes, and requiring only 3 minutes total to solve due to a proper application of model reduction to the control problem. Unreduced optimization took 1 hour for each step. The four images show output poses at times corresponding to four consecutive keyframes (out of 11 total). For comparison, the keyframe is shown in the top-left of each image.

Model reduction is very beneficial to optimal control because it greatly reduces the state and control size. The states u_i are replaced with the reduced states z_i, the control vectors f_i are replaced with the reduced internal forces $\tilde{f}_i$, and Equation 5 is replaced with its reduced version (Equation 6). Although such a reduced optimal control problem only approximates the original problem, its dimensionality is only $rT \ll 3nT$; therefore, the occurrence of local minima is greatly decreased. Optimal reduced forces can be found faster than unreduced forces, because one can rapidly explore the solution space by running many reduced forward simulations and by quickly evaluating the reduced objective gradient [5] (Figure 8). Standard controllers such as the linear-quadratic regulator [37] are impractical with deformable objects as they involve dense $3n \times 3n$ gain matrices. With reduction, however, such control becomes feasible as the gain matrices are now much smaller ($r \times r$). Barbič and Popović [7] exploited such a combination of LQR control and model reduction for real-time tracking of nonlinear deformable object simulations, using minimal ("gentle") forces.

6 Free software for model reduction

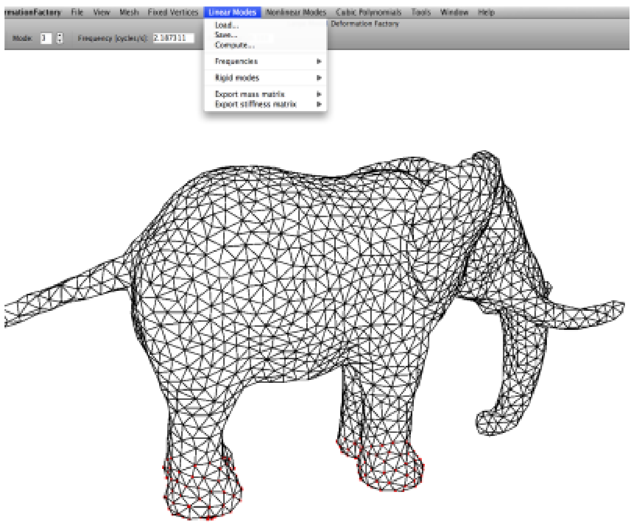
The implementation of [6] (by Jernej Barbič) is freely available on the web (BSD license) at: `http://www.jernejbarbic.com/code`. It includes

1. a precomputation utility to compute linear modes (Equation 1) and modal derivatives (Equation 15), and to construct the simulation basis U (mass-PCA applied to the dataset of Equation 18); optionally, the basis can also be computed from pre-existing simulation data ("POD basis"),

2. a precomputation utility to compute the cubic polynomial coefficients for reduced internal forces $\tilde{f}_{\text{int}}$ and stiffness matrices $\tilde{K}$ (Section 3.1), for isotropic geometrically nonlinear material model (*St. Venant-Kirchhoff material*)

3. an efficient C/C++ library to timestep the reduced model precomputed in the above steps 1., 2. (Equation 6), and an example run-time driver.

The code shares basic classes with *Vega*, a general-purpose simulator for FEM nonlinear 3D deformable objects (including deformable dynamics), also available at the same URL (BSD license).



Mode: 3 Frequency (cycles/s): 2.187311

Figure 9: **Precomputation utility** to compute linear modes, derivatives, basis for large deformations, and cubic polynomial coefficients. Freely available (BSD license) at `http://www.jernejbarbic.com/code` .

7 Deformation warping

As outlined in Section 3, linear modal analysis (Section 2) leads to visible artifacts under large deformations, and these artifacts can be removed by applying model reduction to the nonlinear equations of motion. Deformation warping is an alternative, purely geometric, approach to remedy the same problem. The idea is to keep Equation 2 as the underlying dynamic equation, but to post-process the resulting deformations u using a geometric "filter" that removes large-deformation artifacts. For example, given a tetrahedral mesh, warping establishes a mapping that maps linearized deformations $u \in \mathbb{R}^{3n}$ (away from the rest configuration) to "good-looking" large deformations $q = W(u) \in \mathbb{R}^{3n}$ (also away from the rest configuration). The user is only shown the corrected deformations q. Warping is very robust. For example, twisting deformations, with local rotations as large as several complete 360 degrees cycles, can be easily accommodated. The underlying dynamics, however, is still linear, and this is visible with large-deformation motion as linear deformations essentially follow a sinusoidal curve, $\sin(\omega_i t)$. In contrast, objects simulated using nonlinear methods usually stiffen under large deformations, and therefore spend a smaller percentage of the oscillation cycle time at large deformations.

The idea that modes could be warped to correct large deformation artifacts was first observed by Choi and Ko [11]. They noticed that, by taking the curl of each modal vector, one can derive per-vertex infinitesimal rotations due to the activation of each mode. These rotations can then be integrated in time, resulting in large deformations free of artifacts. Although not surveyed here in detail, their approach is fast, and laid the foundation for other warping methods developed later.

7.1 Rotation-strain coordinate warping

In these notes, we describe a recent efficient flavor of warping, the *rotation-strain coordinate warping* [16]. Let $G_j \in \mathbb{R}^{9\times 3n}$ be the discrete gradient operator of tet j, i.e., deformation gradient of tet j under deformation $u \in \mathbb{R}^{3n}$ equals $I + G_j u$. Decompose the 3×3 matrix $G_j u$ into a symmetric and antisymmetric component,

$$G_j u = \frac{G_j u + (G_j u)^T}{2} + \frac{G_j u - (G_j u)^T}{2}. \tag{19}$$

We can then denote the upper triangle of the symmetric part as $\epsilon_j \in \mathbb{R}^6$, and the skew-vector corresponding to the antisymmetric part as $\omega_j \in \mathbb{R}^3$. We then assemble $[\epsilon_j, \omega_j]$ for all tets into a vector $y(u) \in \mathbb{R}^{9\#tets}$;

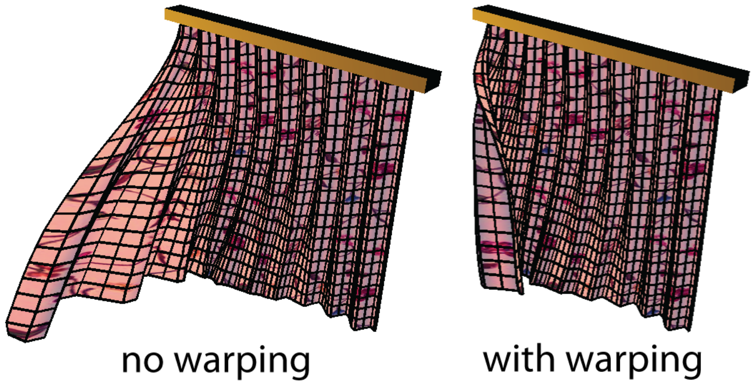


Figure 10: **Warping corrects linearization artifacts under large deformations.**

denote $y_{\epsilon,j} = \epsilon_j$ and $y_{\omega,j} = \omega_j$. Vector y forms the *rotation-strain coordinates* of the mesh. Given a linearized deformation u, we then postulate that we should seek a deformation q so that the deformation gradient inside tet j equals

$$\exp\left(\widetilde{y_{\omega,j}}\right)\left(I + \mathrm{sym}(y_{\epsilon,j})\right). \tag{20}$$

Here, $\mathrm{sym}(x)$ is the 3×3 symmetric matrix corresponding to its upper-triangle $x \in \mathbb{R}^6$, $\tilde{x}$ is the 3×3 skew-symmetric cross-product matrix corresponding to vector $x \in \mathbb{R}^3$ ($\tilde{x}v = x \times v$ for all $v \in \mathbb{R}^3$), and exp is the matrix exponential function [36]. This condition cannot be satisfied for all the tets simultaneously. Instead, given u, we find a deformation q under which the deformation gradients are as close as possible to those given by Equation 20. This can be done by solving the following least-squares problem:

$$\arg\min_{q} \sum_{j=1}^{\#tets} V_j ||I + G_j q - \exp\left(\widetilde{y_{\omega,j}}\right)\left(I + \mathrm{sym}(y_{\epsilon,j})\right)||_F^2, \tag{21}$$

$$\text{subject to} \quad \text{pinned vertices} \tag{22}$$

where $||\ ||_F$ denotes the Frobenius norm of a 3×3 matrix, and V_j is the volume of tet j. The pinned vertices are the vertices where the model is rooted to the ground (boundary conditions). For free-flying objects, a constraint can be formed that keeps the center of mass unmodified. The objective function in Equation 21 is quadratic in q, and can be rewritten as

$$||VGq - b||_2^2, \tag{23}$$

for $V = \mathrm{diag}(\sqrt{V_1}, \sqrt{V_1}, \ldots, \sqrt{V_{\#tets}})$ (each entry repeated 9x), and where $G \in \mathbb{R}^{9\#tets \times 3n}$ is the gradient matrix assembled from all G_j. The nine-block of vector $b \in \mathbb{R}^{9\#tets}$ corresponding to tet j, expressed as a row-major 3×3 matrix, equals

$$b_j = \sqrt{V_j}\left(\exp(\widetilde{y_{\omega,j}})(I + \mathrm{sym}(y_{\epsilon,j})) - I\right). \tag{24}$$

In a typical tet mesh, there are more tets than vertices, therefore, the optimization problem is overconstrained. The minimization can be performed via Lagrange multipliers, by solving

$$\begin{bmatrix} L & d \\ d^T & 0 \end{bmatrix} \begin{bmatrix} q \\ \lambda \end{bmatrix} = \begin{bmatrix} (VG)^T b \\ 0 \end{bmatrix}, \tag{25}$$

where $L = G^T V^2 G$, and where d corresponds to the pinned input vertices (note: an implementation can simply remove the rows-columns from L; this is equivalent). Matrix L is called the *discrete Laplacian of the mesh*. It only depends on the input mesh geometry, and not on U or u. The system matrix in Equation 25 is sparse and constant, and can be pre-factored, so warping can be performed efficiently at runtime.

7.2 Warping for triangle meshes

Triangle meshes are commonly employed in computer graphics, say, for simulation of thin shells and cloth. Such physical systems also produce the mass matrix M and stiffness matrix K. Therefore, the small deformation analysis (Equation 2) and model reduction (as in Equation 3) apply also to such problems. In order to apply warping, however, we must define deformation gradients for each triangle. As observed by Sumner and Popović [38], the three vertices of a triangle before and after deformation do not fully determine the affine transformation since they do not establish how the space perpendicular to the triangle deforms. They resolve this issue by adding a (fictitious) fourth vertex v_4,

$$v_4 = v_1 + \frac{(v_2 - v_1) \times (v_3 - v_1)}{\sqrt{|(v_2 - v_1) \times (v_3 - v_1)|}}, \tag{26}$$

where v_1, v_2, v_3 are the triangle vertices. Vertices v_1, v_2, v_3, v_4 define a tetrahedron. Let v_1', v_2', v_3' denote the deformed vertex positions; then, we can use (26) to compute the deformed fictitious vertex v_4'. The deformation gradient F for the triangle equals

$$F = \begin{bmatrix} v_2' - v_1' & v_3' - v_1' & v_4' - v_1' \end{bmatrix} \begin{bmatrix} v_2 - v_1 & v_3 - v_1 & v_4 - v_1 \end{bmatrix}^{-1}. \tag{27}$$

Given the deformation gradient, warping then proceeds in the same way as described for tetrahedral meshes in previous sections. A more principled version of triangle mesh warping has been presented by [12].

8 Acknowledgements

This work was sponsored by the National Science Foundation (CAREER-53-4509-6600). I would like to thank Eftychios Sifakis and Yili Zhao for helpful suggestions.

References

[1] V. Akcelik, J. Bielak, G. Biros, I. Epanomeritakis, A. Fernandez, O. Ghattas, E. J. Kim, J. Lopez, D. O'Hallaron, T. Tu, and J. Urbanic. High-resolution forward and inverse earthquake modeling on terascale computers. In *Proceedings of ACM/IEEE SC2003*, 2003.

[2] S. S. An, T. Kim, and D. L. James. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. on Graphics*, 27(5):165:1–165:10, 2008.

[3] D. Baraff and A. P. Witkin. Large Steps in Cloth Simulation. In *Proc. of ACM SIGGRAPH 98*, pages 43–54, July 1998.

[4] J. Barbič. *Real-time Reduced Large-Deformation Models and Distributed Contact for Computer Graphics and Haptics*. PhD thesis, Carnegie Mellon University, Aug. 2007.

[5] J. Barbič, M. da Silva, and J. Popović. Deformable object animation using reduced optimal control. *ACM Trans. on Graphics (SIGGRAPH 2009)*, 28(3):53:1–53:9, 2009.

[6] J. Barbič and D. L. James. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. on Graphics*, 24(3):982–990, 2005.

[7] J. Barbič and J. Popović. Real-time control of physically based simulations using gentle forces. *ACM Trans. on Graphics (SIGGRAPH Asia 2008)*, 27(5):163:1–163:10, 2008.

[8] J. Barbič and Y. Zhao. Real-time large-deformation substructuring. *ACM Trans. on Graphics (SIGGRAPH 2011)*, 30(4):91:1–91:7, 2011.

[9] B. Bickel, M. Baecher, M. Otaduy, W. Matusik, H. Pfister, and M. Gross. Capture and modeling of non-linear heterogeneous soft tissue. *ACM Trans. on Graphics (SIGGRAPH 2009)*, 28(3):89:1–89:9, 2009.

[10] J. N. Chadwick, S. S. An, and D. L. James. Harmonic Shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics*, 28(5):1–10, 2009.

[11] M. G. Choi and H.-S. Ko. Modal Warping: Real-Time Simulation of Large Rotational Deformation and Manipulation. *IEEE Trans. on Vis. and Comp. Graphics*, 11(1):91–101, 2005.

[12] M. G. Choi, S. Y. Woo, and H.-S. Ko. Real-Time Simulation of Thin Shells. *Eurographics 2007*, pages 349–354, 2007.

[13] L. Daniel. Private correspondence with Prof. Luca Daniel, MIT.

[14] K. K. Hauser, C. Shen, and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Proc. of Graphics Interface*, pages 247–256, 2003.

[15] J. Huang, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. An efficient large deformation method using domain decomposition. *Computers & Graphics*, 30(6):927 – 935, 2006.

[16] J. Huang, Y. Tong, K. Zhou, H. Bao, and M. Desbrun. Interactive shape interpolation through controllable dynamic deformation. *IEEE Trans. on Visualization and Computer Graphics*, 17(7):983–992, 2011.

[17] D. L. James, J. Barbič, and D. K. Pai. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (SIGGRAPH 2006)*, 25(3), 2006.

[18] D. L. James and D. K. Pai. DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Trans. on Graphics*, 21(3):582–585, 2002.

[19] D. L. James and D. K. Pai. Real Time Simulation of Multizone Elastokinematic Models. In *IEEE Int. Conf. on Robotics and Automation*, pages 927–932, 2002.

[20] D. L. James and D. K. Pai. BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models. *ACM Trans. on Graphics*, 23(3):393–398, 2004.

[21] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai. Staggered Projections for Frictional Contact in Multibody Systems. *ACM Transactions on Graphics*, 27(5):164:1–164:11, 2008.

[22] A. E. Kerdok, S. M. Cotin, M. P. Ottensmeyer, A. M. Galea, R. D. Howe, and S. L. Dawson. Truth cube: Establishing physical standards for soft tissue simulation. *Medical Image Analysis*, 7(3):283–291, 2003.

[23] T. Kim and D. James. Skipping steps in deformable simulation with online model reduction. *ACM Trans. on Graphics (SIGGRAPH Asia 2009)*, 28(5):123:1–123:9, 2009.

[24] T. Kim and D. James. Physics-based character skinning using multi-domain subspace deformations. In *Symp. on Computer Animation (SCA)*, pages 63–72, 2011.

[25] C. L. Lawson and R. J. Hanson. *Solving Least Square Problems*. Prentice Hall, Englewood Cliffs, NJ, 1974.

[26] R. Lehoucq, D. Sorensen, and C. Yang. ARPACK Users' Guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods. Technical report, Comp. and Applied Mathematics, Rice Univ., 1997.

[27] J.-R. Li. *Model Reduction of Large Linear Systems via Low Rank System Gramians*. PhD thesis, Massachusetts Institute of Technology, 2000.

[28] R.-C. Li and Z. Bai. Structure preserving model reduction using a Krylov subspace projection formulation. *Comm. Math. Sci.*, 3(2):179–199, 2005.

[29] J. L. Lumley. The structure of inhomogeneous turbulence. In A.M.Yaglom and V.I.Tatarski, editors, *Atmospheric turbulence and wave propagation*, pages 166–178, 1967.

[30] A. McNamara, A. Treuille, Z. Popović, and J. Stam. Fluid control using the adjoint method. *ACM Trans. on Graphics (SIGGRAPH 2004)*, 23(3):449–456, 2004.

[31] J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Symp. on Computer Animation (SCA)*, pages 175–181, 2002.

[32] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proc. of ACM SIGGRAPH 89)*, 23(3):215–222, 1989.

[33] M. Rewienski. *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems*. PhD thesis, Massachusetts Institute of Technology, 2003.

[34] A. Safonova, J. Hodgins, and N. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. on Graphics (SIGGRAPH 2004)*, 23(3):514–521, 2004.

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

[36] R. B. Sidje. Expokit: A Software Package for Computing Matrix Exponentials. *ACM Trans. on Mathematical Software*, 24(1):130–156, 1998. www.expokit.org.

[37] R. F. Stengel. *Optimal Control and Estimation*. Dover Publications, New York, 1994.

[38] R. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. on Graphics (SIGGRAPH 2004)*, 23(3):399–405, 2004.

[39] A. Treuille, A. Lewis, and Z. Popović. Model reduction for real-time fluids. *ACM Trans. on Graphics*, 25(3):826–834, 2006.

[40] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. *ACM Trans. on Graphics (SIGGRAPH 2003)*, 22(3):716–723, 2003.

[41] M. Wicke, M. Stanton, and A. Treuille. Modular bases for fluid dynamics. *ACM Trans. on Graphics*, 28(3):39:1–39:8, 2009.

[42] C. Wojtan, P. J. Mucha, and G. Turk. Keyframe control of complex particle systems using the adjoint method. In *Symp. on Computer Animation (SCA)*, pages 15–23, Sept. 2006.