Subspace Integration with Local Deformations

David Harmon and Denis Zorin
New York University

Abstract

Subspace techniques greatly reduce the cost of nonlinear simulation by approximating deformations with a small custom basis. In order to represent the deformations well (in terms of a global metric), the basis functions usually have global support, and cannot capture localized deformations. While reduced-space basis functions can be localized to some extent, capturing truly local deformations would still require a very large number of precomputed basis functions, significantly degrading both precomputation and online performance. We present an efficient approach to handling local deformations that cannot be predicted, most commonly arising from contact and collisions, by augmenting the subspace basis with custom functions derived from *analytic* solutions to static loading problems. We also present a new cubature scheme designed to facilitate fast computation of the necessary runtime quantities while undergoing a changing basis. Our examples yield a two order of magnitude speedup over full-coordinate simulations, striking a desirable balance between runtime speeds and expressive ability.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Links: DL PDF CODE

1 Introduction

The simulation of nonlinear deformations is expensive. In particular, simulations involving 3D volumetric elements are particularly costly, and the resolution of such simulations in computer graphics applications often has to be kept low for performance reasons. To address this concern, many methods use model reduction to construct a set of basis vectors which span the space of *interesting* deformations. These bases allow the simulation of linear and nonlinear deformation with far fewer degrees-of-freedom (DOFs) than a full finite element simulation, while maintaining high discretization resolution needed for graphics applications.

Constructing a good basis is difficult without *a priori* knowledge of the system dynamics. Common approaches yield *global* bases, with non-zero displacements of all mesh vertices, e.g., low-frequency eigenmodes. Most simulations of interest result in deformations with a significant smooth global component, which a globally supported smooth basis captures well.

However, this approach excludes interesting phenomena involving spatially-localized, rapidly varying components of deformations, such as those induced by collisions and persistent contact. Representing these deformations with global bases would require such a high number of DOFs that any computational advantage would quickly be negated.

Figure 1: *Global subspace methods cannot capture fine scale behavior, such as these indentations where the Cheburashka mesh is hit by a series of projectiles.*

In this paper, we present a simple method for dynamically augmenting a basis with vectors that allow representation of local deformations, such as those resulting from collisions and contact. Basis augmentation for resolving detail is well-known in simulation applications: in its simplest form, it means simply refining and de-refining a finite element mesh, effectively adding and removing small-scale basis functions. Our goal is to demonstrate how the same general idea can be applied to bases adapted to reduced model simulations, which uses orders of magnitude fewer basis functions.

Overview. The basic idea of our approach is to decompose elastic deformations of an object into two parts: the global smooth deformation and localized deformations near loaded surface points. The former, as confirmed by numerous previous studies, is approximated well by a small basis of precomputed modal deformations. To deal with the latter we approximate the deformation near the load by a localized version of a precomputed linear deformation in response to a force applied at the point of interest. As the number of loaded regions in a mesh is typically small, the number of active local functions needed for the simulation is also likely to be small.

Dynamically introducing additional basis functions for local deformations poses a problem with precomputation: unlike global deformation modes, a separate basis function would have to be precomputed for every surface point, or small groups of points, which results in a very high cost in precomputation and storage. On the other extreme, using completely local basis functions (e.g., p.w. linear functions) requires adding a large number of additional DOFs. Instead, we observe that point-load deformations decay quickly, and for most surface points, the dominant part can be well approximated by on-demand analytic functions. These functions are fast to construct and do not require additional precomputation (§4.1); yet far fewer are required compared to the finite element basis.

Prior work in subspace integration has taken great pains to maintain a computational complexity that is a small power of the number of basis vectors. This is made possible by significant precomputation that can easily be ruined by the introduction of dynamic bases (§5.1). We show that for the problem of local deformations we can maintain the reduced complexity needed for interactive simulations in two ways: our subspace DOFs are output-sensitive to the number of loaded regions in the model, which is in most cases small (§4.2), and we use a novel cubature sampling strategy that takes advantage of the structure of loaded surfaces (§5.2).

We demonstrate dynamically generated subspace bases for deformations arising from transient collisions and persistent contact. Our local subspace addition introduces only a modest computational burden, yielding a two order of magnitude speedup speedup over full-coordinate simulations for our examples (§7).

2 Related work

Subspace integration techniques have a long history in engineering [Nickell 1976; Bathe and Gracewski 1981], with modal analysis a popular method for construction of a reduced basis [Thomson 2004]. Later, these ideas were introduced to the graphics community by Pentland and Williams [1989], and extended to the nonlinear regime by Barbič and James [2005], which built on similar work from engineering [Idelsohn and Cardona 1985b].

Moving beyond linear modal analysis, Barbič and James [2005] expanded the concept of *modal derivatives* to represent large deformations and experimented with custom bases derived from sample forces and offline static solutions. Example-based shape synthesis has been a popular research topic [Koyama et al. 2012; Martin et al. 2011], however these methods are dependent on the quality of the training data provided.

Kim and James [2011] perform a domain decomposition with modal analysis on each domain, which are then coupled together at boundaries, allowing restricting deformations to precomputed domains. Truly local displacements, however, cannot be represented unless decomposition is performed at a prohibitively fine level. Barbič and Zhao [2011] build a hierarchy out of mesh components, with impulses transferred through the (small) interfaces down the hierarchy. This allows somewhat local displacements, but again limited to the mesh partitioning and specific mesh types that admit this type of substructuring.

Idelsohn and Cardona [1985a] compute new basis vectors dynamically, while Kim and James [2009] perform online model reduction to adaptively sample a precomputed basis. While not intended for local deformations, we make use of several ideas from this paper to dynamically adapt a basis on-the-fly to external stimulii.

Ogot et al. [1996] proposed a hybrid simulation that uses rigid body dynamics for free flight and explicit finite elements during the contact phase. At a high level this is similar to our approach, but we are not limited to purely rigid motion and do not resort to full finite elements during contact, avoiding the limitations of both ends.

Adaptive refinement in finite element simulations has been an actively studied topic in both engineering and graphics [Debunne et al. 2001; Wu et al. 2001; Grinspun et al. 2002]. In contrast to these methods, which begin with a full-coordinate basis and then refine/coarsen, we begin with a *modal* basis and then dynamically build local modes from displacement fields. Our approach requires significantly fewer basis functions (Fig. 3), and is not as dependent on the underlying mesh geometry.

An et al. [2008] developed a cubature scheme for approximating integrals across a mesh using a small set of representative elements. Our work uses cubature as well, but instead of an expensive precomputation, we construct our representative set and cubature weights dynamically.

In graphics, simulating elastic objects independent of an underlying mesh has been an active topic of research. Point-based methods [Gerszewski et al. 2009] as well as meshless methods [Faure et al. 2011] are popular alternatives. Previous work in precomputed Green's functions [James and Pai 2003], analytic solutions, such as the Boussinesq solution [Pauly et al. 2004], and fundamental solutions to Navier's equations [James and Pai 1999] are all similar in spirit to our method: using precomputed (or analytic) response functions to accelerate online computation.

3 Subspace integration

The second order Euler-Lagrange equations describe the motion of a deformable object:

$$\mathbf{M}\ddot{\mathbf{u}} + D(\mathbf{u}, \dot{\mathbf{u}}) + R(\mathbf{u}) = \mathbf{f}, \tag{1}$$

where $\mathbf{M}$ is the system mass matrix, $\mathbf{u}$ is the vector of displacements, D, R, and $\mathbf{f}$ denote damping, internal, and external forces, respectively, and dots denote time derivatives.

In model reduction, the displacement vector is represented in a reduced set of coordinates,

$$\mathbf{u} = \mathbf{U}\mathbf{q},$$

where $\mathbf{U} \in \mathbb{R}^{3n,r}$ is a matrix with columns corresponding to the vectors of the reduced displacement basis and $\mathbf{q} \in \mathbb{R}^r$ is the vector of reduced coordinates. The dimension of the subspace r is chosen to be much less than $3n$. Substituting $\mathbf{u} = U\mathbf{q}$ into Eqn. (1) we arrive at the reduced equations of motion, approximating (1) with far fewer degrees-of-freedom:

$$\tilde{\mathbf{M}}\ddot{\mathbf{q}} + \tilde{D}(\mathbf{q}, \dot{\mathbf{q}}) + \tilde{R}(\mathbf{q}) = \tilde{\mathbf{f}}, \tag{2}$$

where a tilde denotes a reduced quantity, obtained by projecting into the chosen subspace: $\tilde{\mathbf{M}} = \mathbf{U}^T M \mathbf{U}$, $\tilde{D}(\mathbf{q}, \dot{\mathbf{q}}) = U^T D(\mathbf{U}\mathbf{q}, \mathbf{U}\dot{\mathbf{q}})$, $\tilde{R}(\mathbf{q}) = \mathbf{U}^T R(\mathbf{U}\mathbf{q})$ and $\tilde{\mathbf{f}} = \mathbf{U}^T \mathbf{f}$.

All good reduced bases have *global* support, as most significant elastic deformations tend to be global. This comes at the cost of being unable to represent local deformations, which not only improve visual quality (Fig. 1), but can also influence the dynamic behavior (Fig. 2).

3.1 Augmenting basis vectors

For any given simulation step, any $r' < r$ subset of a precomputed basis may be chosen for simulation, temporarily disabling representation of the remaining $r - r'$ basis vectors. Kim and James [2009] use this observation to precompute a basis and decide on-line which basis vectors to consider based on error metrics. We note that this approach could be reversed: we can think of the pre-selected r basis vectors as a subset of a larger, unknown basis $r' > r$. Theoretically, there are an infinite number of such unrepresented vectors ($r' = \infty$), but during runtime we have the advantage that we can augment the basis with those vectors which are most likely to capture the physics of the system, rather than adding basis vectors from a precomputed set.

We denote the matrix of r precomputed global basis vectors as $\mathbf{V}$ and the matrix of additional s local deformation basis vectors as $\mathbf{W}$, so that $\mathbf{U} = [\mathbf{V}\ \ \mathbf{W}] \in \mathbb{R}^{3n,r'}$ with $r' = r + s$. $\mathbf{V}$ will be precomputed using existing methods (e.g., [Pentland and Williams 1989] or

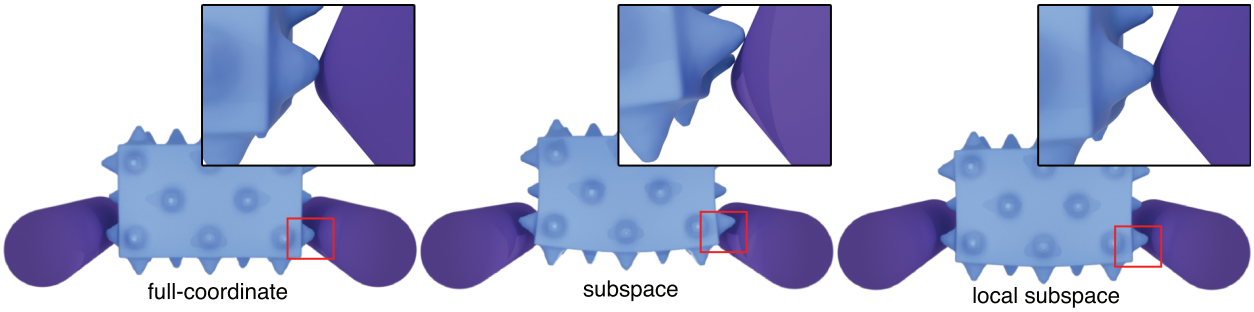


Figure 2: *Local deformations can also significantly effect the simulation dynamics. A spikey box is dropped under gravity between two unfortunately-spaced cylinders. A global subspace cannot represent the local deformations needed to allow the box to slip between the obstacles. The local subspace method better approximates the overall behavior of the full-coordinate simulation.*

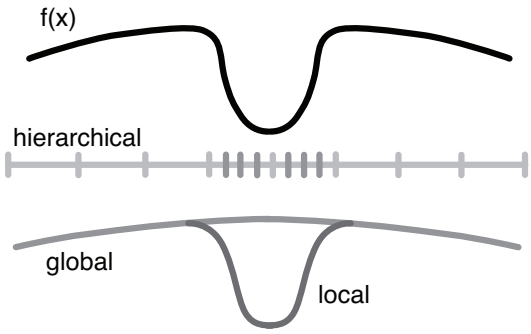


Figure 3: *Representing the $1D$ function $f(x)$ using a hierarchical or subdivided basis would require an excessive number of basis functions, whereas a global basis combined with a shape-aware local basis can accurately capture the configuration with only a few basis functions.*

[Barbič and James 2005]). $\mathbf{W}$ will be dynamically updated based on the forces present in the system. Our focus is on deformations due to local forces, hence in the next section we show how to compute basis vectors in $\mathbf{W}$ in response to locally applied loads.

4 Local basis functions

We consider the specific case of augmenting a basis in order to represent displacements from surface loads on the boundary of an object. This includes many interactive tools as well as deformation arising from collisions and contact. The only restriction we place on these loads is that the resulting displacements are small enough to be described by linear elasticity. Note, however, that this displacement is relative to existing modal deformations, so that in aggregate any node may experience quite large displacements.

4.1 Generating local basis vectors

The most direct way to augment a basis with local basis functions is to use a local finite element basis, e.g., adding hat functions at the nodes in the region of interest. The main problem of such a strategy is the number of basis functions that would be required to represent even modest displacements (see Fig. 3). Not only would three degrees of freedom be required per displaced surface node, but interior nodes must also be included for a reasonable representation of displacement dynamics. In many cases these additional bases have to cover a significant fraction of the mesh for faithful representation of deformations. Instead, we observe that the loads of interest are often localized, thus, the *load* can be approximated well by a small number of local functions, while the deformation *response* can be approximated by the solutions of static elasticity problems with point loads. Our idea is to adapt solutions to such problems, in the cases when these can be obtained analytically, for use as additional basis functions.

Solutions to boundary load problems. Let $\mathbf{f}$ be a given load on a single vertex, i.e., zero everywhere except for the vertex of interest. The exact solution for a given load $\mathbf{f}$ of a linear elasticity problem is the solution $\mathbf{u}$ to the following linear equations:

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \tag{3}$$

where $\mathbf{K}$ is the $3n \times 3n$ Jacobian of R (Eqn. (1)), the FEM stiffness matrix. As this is the static solution for the given load, we expect that $\mathbf{u}$ would form a good basis vector in $\mathbf{W}$ to represent displacement in response to loads in the direction of $\mathbf{f}$, similar in manner to Barbič and James [2005], which utilizes static solutions to external forces to construct basis vectors in an off-line process.

Analytic solutions. Eqn. (3), can be factored during precomputation and solved at runtime in $O(n)$ time, while caching basis vectors should they be needed again. Nevertheless, for meshes that are very large ($n \gg r^4$), this expense can cause considerable slowdown. To ease this burden we adapt analytic solutions to static loading problems on a half-space to arbitrary volumes.

Analytic solutions under linear elasticity for a loaded half-space have been well-studied since Boussinesq, and can be found in standard texts on linear elasticity and contact mechanics (e.g., Johnson [1987]). In graphics, Pauly et al. [2004] used the Boussinesq solution for describing contact in point clouds. The solution for a normal load $\mathbf{f}$ on the half-space bounded by the xz plane is:

$$u_x = \frac{f}{4\pi G}\left(\frac{xy}{\rho^3} - (1-2\nu)\frac{x}{\rho(\rho+y)}\right) \tag{4}$$

$$u_y = \frac{f}{4\pi G}\left(\frac{y^2}{\rho^3} + \frac{2(1-\nu)}{\rho}\right) \tag{5}$$

$$u_z = \frac{f}{4\pi G}\left(\frac{zy}{\rho^3} - (1-2\nu)\frac{z}{\rho(\rho+y)}\right), \tag{6}$$

where $\rho = \sqrt{x^2+y^2+z^2}$ and G and ν are the shear modulus and Poisson ratio for the material. The actual basis vector is constructed as a linear approximation to the analytic expression Eqn. (4)-(6) by aligning a coordinate frame at the vertex with the surface normal and sampling it at the volume vertices. These functions are singular at the origin (the point around which the basis is being constructed); we used a "regularized" version, with values at the central vertex computed as a weighted average of the displacements of the neighboring element centroids.

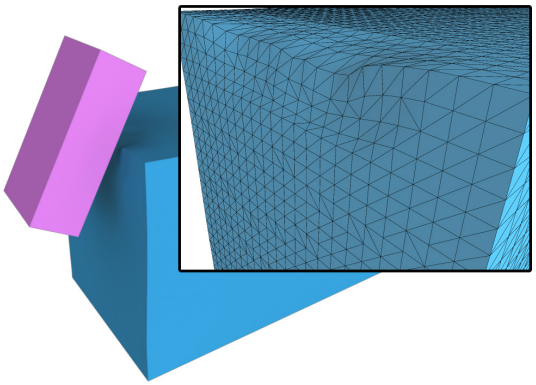


Figure 4: *The analytic function for a half-space adapts to non-smooth geometry, such as this box edge being pressed by a block.*

A basis vector constructed for a force $\mathbf{f}$ can represent response to that specific load, yet we wish our basis vectors to represent more general dynamics. Hence, we will compute three basis vectors: $\mathbf{b}_i^x, \mathbf{b}_i^y$, and $\mathbf{b}_i^z$, where $\mathbf{b}_i^j$, for $j = x, y, z$, is the analytic solution under loading $\mathbf{f}$, with $\mathbf{f}_i^j$ non-zero in the j-th DOF of vertex i and zero elsewhere. For completeness, the analytic solution to a tangential load of magnitude f in the x-direction is as follows:

$$u_x = \frac{f}{4\pi G}\left(\frac{1}{\rho} + \frac{x^2}{\rho^3} + (1-2\nu)\left[\frac{1}{\rho+y} - \frac{x^2}{\rho(\rho+y)^2}\right]\right) \quad (7)$$

$$u_y = \frac{f}{4\pi G}\left(\frac{xy}{\rho^3} - (1-2\nu)\frac{xy}{\rho(\rho+y)^2}\right) \quad (8)$$

$$u_z = \frac{f}{4\pi G}\left(\frac{zy}{\rho^3} + (1-2\nu)\frac{x}{\rho(\rho+y)}\right). \quad (9)$$

Linear combinations of this triplet can represent an arbitrary loading of vertex i, and overall, we find that these analytic functions work well for general shapes, including non-smooth geometry (Fig. 4).

Co-rotated basis. The above-defined basis functions are sufficient for representing local displacements relative to small global deformations. However, practical simulations often require large, non-linear deformations, such as in Barbič and James [2005], which introduced a model reduction method for these types of dynamics. To be able to use our basis functions in this context, we need to rotate the frame so that local displacements are relative to these deformations. We follow the development of co-rotated basis functions for a finite element basis, such as in Hauth and Strasser [2004].

We extract the rigid rotation of the one-ring of the vertex central to the basis by solving the Procrustes problem and computing the SVD of the 3×3 matrix $\mathbf{X}^T\mathbf{X}$, where each row of $\mathbf{X}$ is $\mathbf{x}_j - \mathbf{x}_i$ for each neighbor j of vertex i. We store this rotation $\mathbf{T}_i = \mathbf{U}\mathbf{V}^T$ (where $\text{SVD}(\mathbf{X}\mathbf{X}^T) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$) as associated with the i-th basis vector, and co-rotate any quantities before projection:

$$\mathbf{q}_j = \mathbf{b}_i \mathbf{T}_i^{-1} \mathbf{u}_j.$$

4.2 Grouping basis functions

Although three elasticity-based basis functions per loaded surface node is far more efficient than using hat basis functions directly, for large contact regions, or equivalently a high-resolution mesh, the number of basis functions can grow large. One primary benefit of subspace simulation is the system in Eqn. 2 has dimension r and can be solved quite efficiently. We investigate options for grouping nearby basis functions when the number of loaded surface nodes exceeds Cr, for a constant C, without sacrificing a noticeable amount of visual fidelity.

Our approach is to cluster the loaded nodes into patches of uniform size, and retain three DOFs per patch. Let S be a loaded surface patch. We define three basis functions for this patch as

$$\mathbf{b}_S^\alpha(p) = \int_{\mathbf{v}\in S} \mathbf{b}_\mathbf{v}^\alpha(\mathbf{p}) dS$$

where $\mathbf{b}_S^\alpha$, $\alpha = x, y, z$, is the basis function for coordinate α for the patch, $\mathbf{v}$ is a point on the patch and $\mathbf{p}$ is a 3D point where we evaluate $\mathbf{b}_S$. Physically speaking, for sufficiently flat patches S, these basis functions correspond to unit load over an area, instead of a single point.

The discrete approximation of this integral is simply

$$\mathbf{b}_i^\alpha = \sum_{j\in S_i} \mathbf{b}_j^\alpha A_j. \quad (10)$$

Here S_i is the set of vertices in a surface region and A_j is the barycentric surface area around node j. Replacing per-node functions $\mathbf{b}_j$ with a sparser set of $\mathbf{b}_i$ amounts to subsampling of local deformations, so the tradeoff in the choice of C is between performance and fidelity of the solution. We use $C = 3$ for all our examples.

Partitioning regions. To arrive at the sets S_i, we partition the entire set of *active* nodes into Cr regions. Nodes are considered active if undergoing an external force, or if either velocity or displacement of the *local basis DOF* that the node is grouped in are greater than ϵ. We use $\epsilon = 10^{-4}\gamma$, where γ is the radius of the sphere enclosing the mesh, and divide by Δt when comparing against the DOF velocity.

First, we partition the set of active surface nodes into connected components, typically this number is less than r (if it is not, then we define at least one basis function per component). Each set is further partitioned into domains of approximately uniform size using k-means clustering with $k = \lceil rA_i/A \rceil$, where A is the total surface area of $S = \sum S_i$ under consideration and A_i is the surface area of this region. This way, each set is allocated basis vectors proportional to its area contribution. We create three basis vectors per sub-region, which, in addition to the original r global vectors results in a total of $(3C+1)r$ basis vectors.

Projecting basis vectors. Our basis vectors, and the regions they affect, are changing over time. To ensure consistent dynamics, we must project subspace vectors $\mathbf{q}$ and $\dot{\mathbf{q}}$ into the new basis each time the basis changes. If the modified basis coefficient matrix is $\bar{U}$, the new DOFs $\bar{\mathbf{q}}$ minimize

$$\|\bar{\mathbf{U}}^T\bar{\mathbf{q}} - \mathbf{U}^T\mathbf{q}\|^2.$$

This results in the normal equations for new DOFs

$$\bar{\mathbf{q}} = \bar{\mathbf{U}}\mathbf{U}^T\mathbf{q},$$

and similar for $\dot{\bar{\mathbf{q}}}$. Timesteps must be small enough to capture the local, small-scale dynamics, which also ensures sufficient temporal coherence for minimal energy loss in this projection.

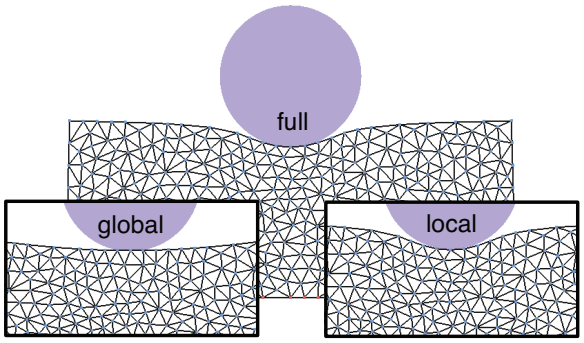


Figure 5: *We press a circle into a 2d gelatinous block. The full simulation produces a local displacement which cannot be replicated with a small number of subspace vectors. Our local subspace augments the global basis to accurately reproduce this behavior.*

5 Computing runtime quantities

The dominant cost in geometrically nonlinear simulation is the computation of the stiffness matrix $\mathbf{K}(\mathbf{u})$, which must be re-evaluated with changes in configuration.

In order to obtain significant speedups, previous work expressed the entries of $\tilde{\mathbf{K}}$, the reduced $r \times r$ stiffness matrix, as cubic polynomials of the reduced coordinates $\mathbf{q}$ [Barbič and James 2005]. This reduced runtime computation of the stiffness matrix from $O(n^2)$ to $O(r^4)$—a significant improvement that often allows for interactive simulation. Later work on cubature schemes reduced this even further to $O(r^2)$ for stiffness matrix construction (solving the linear system of equations remains $O(r^3)$). For the local subspace method of §4 to be a competitive alternative to full FEM simulations, it too must offer runtime complexities of $O(r^4)$ or better.

5.1 Error and computational complexity

Fig. 5 compares using full coordinate basis, global subspace basis, and global subspace augmented with a local basis for a simple test of a circle pressed into a gelatinous block. While improving appearance, dynamically generated basis vectors introduce a technical problem: polynomial coefficients or cubature weights can no longer be precomputed and the full stiffness matrix, mass matrix, and force vector must be computed, then projected into the reduced subspace each timestep:

$$\tilde{\mathbf{K}} = \mathbf{U}^T\mathbf{K}\mathbf{U} = \begin{pmatrix} \mathbf{V}^T\mathbf{K}\mathbf{V} & \mathbf{W}^T\mathbf{K}\mathbf{V} \\ \mathbf{V}^T\mathbf{K}\mathbf{W} & \mathbf{W}^T\mathbf{K}\mathbf{W} \end{pmatrix} \quad (11)$$

$$\tilde{\mathbf{M}} = \mathbf{U}^T\mathbf{M}\mathbf{U} = \begin{pmatrix} \mathbf{V}^T\mathbf{M}\mathbf{V} & \mathbf{W}^T\mathbf{M}\mathbf{V} \\ \mathbf{V}^T\mathbf{M}\mathbf{W} & \mathbf{W}^T\mathbf{M}\mathbf{W} \end{pmatrix} \quad (12)$$

$$\tilde{\mathbf{F}} = \mathbf{U}^T\mathbf{F} = \begin{pmatrix} \mathbf{V}^T\mathbf{F} \\ \mathbf{W}^T\mathbf{F} \end{pmatrix}. \quad (13)$$

While the top-left block of Eqn. 11, $\mathbf{V}^T\mathbf{K}\mathbf{V}$, remains the same and can continue to take advantage of precomputation, the top-right and bottom-right blocks rely on the changing basis vectors in $\mathbf{W}$ (the matrix is symmetric so the bottom-left block can be ignored).

To combat this issue, we reduce computations by building on previous work in cubature optimization. Unlike previous work, we use sampling and construct cubature weights *dynamically*, to achieve $O(r^3)$ runtimes for the computation of Eqns. (11)–(13).

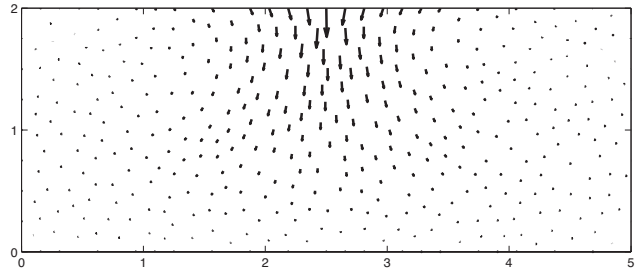


Figure 6: *The basis generated from Eqn.* (5)*. The mesh is the 2d mesh shown in Fig. 5, with the load applied to the center vertex at the top with coordinates* $(2.5, 2)$*. Each vector is the displacement for the vertex at that location.*

5.2 Cubature sampling

To begin, we note that Eqn. (13) (and similarly, Eqns. (11)–(12)) can be written as the sum of contributions over the mesh elements $\mathcal{E}$:

$$\tilde{\mathbf{F}} = \sum_{e \in \mathcal{E}} \mathbf{U}_e^T \mathbf{f}_e. \quad (14)$$

$\mathbf{f}_e$ is the 12-vector of forces for a single tetrahedral element and $\mathbf{U}_e^T$ is the $r' \times 12$ sub-block of $\mathbf{U}^T$ for the respective element nodes. Rather than computing the full $3n$ vector $\mathbf{F}$ and projecting as in Eqn. (13), we will reduce the set of elements over which the above summation is required.

Basis truncation. Fig. 6 shows a 2d basis function from the analytic solution. We can see clearly that the influence of the basis decreases rapidly away from the loaded surface point. This suggests that the basis can be *truncated*, by ignoring the influence of elements outside some radius of interest, κ_o, and interpolating smoothly to zero for distances between κ_o and some inner radius, e.g., $\kappa_i = 0.9\kappa_o$. κ_o could be chosen to remove elements with contribution less than some magnitude, but we choose a radius that contains $O(r^2)$ elements, noting that even for high resolution meshes, the influence decays faster than this radius, except in the case of incredibly stiff materials, which exhibit no local displacements anyway (e.g., steel). Let d be the average density ($d = 1/(\sum V_i/n_e)$, or the inverse of average element volume). Then $\kappa_o = \sqrt[3]{r^2/d}$. This chooses κ_o so that $O(r^2)$ elements are in the sampling sphere.

The first r entries of $\tilde{\mathbf{F}}$ are computed via precomputed polynomial coefficients or cubature sampling, as in prior work. After truncation, however, most elements in $\mathcal{E}$ will have no contribution to the tail s entries of the summation in Eqn. (14), since their respective sub-blocks in the matrix $\mathbf{W}$ are now zero. These elements can thus be excluded from the summation.

Even after truncation, the computation of Eqns. (11)–(13) can dominate runtime. Following ideas developed in An et al. [2008], we can only compute the force (stiffness, mass) on a subset of elements, and with the appropriate weighting accurately approximate the exact reduced force vector:

$$\tilde{\mathbf{F}}_{r:r'} = \mathbf{W}^T\mathbf{F} \approx \sum_{c \in \mathcal{C}} w_c \mathbf{W}_c^T \mathbf{f}_c. \quad (15)$$

Here, $\mathcal{C} \subset \mathcal{E}$ is a set of representative elements drawn from the full set, and $w_c \in \mathbb{R}$ is a weighting that should be chosen to minimize error and increase stability.

In An et al. [2008], the set $\mathcal{C}$ and weightings are obtained as a nested preprocess that iteratively attempts to find the best next sam-

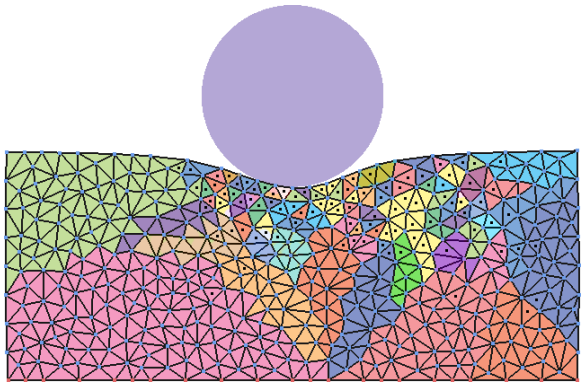


Figure 7: *Sampling pattern for a simple 2d example. Contiguous, same-colored triangles are grouped together in sets $\mathcal{E}_i$. Black dots indicate key elements in the set $\mathcal{C}$.*

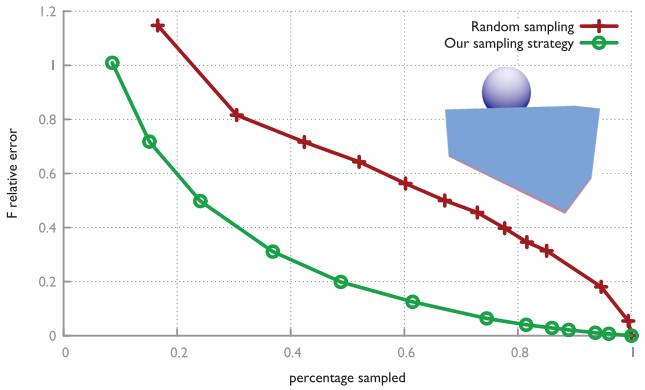


Figure 8: *We plot the relative error in forces for different sampling amounts, given as a fraction of total elements (i.e., 1 means all elements are sampled and the error is 0 for any sampling strategy). We compare our guided sampling strategy and a strategy which randomly chooses elements out of those within a given radius. We compute weights in the same manner for both strategies.*

ple element, and then solves an optimization problem to minimize the error. This process is terminated when the error falls below a given threshold. Unfortunately, since $\mathbf{W}$ is generated at runtime, we cannot take advantage of this preprocessing. We will consider the process of selecting sample elements and computing weights separately.

Sampling elements. Solving a large optimization problem during runtime is inpractical. At the other extreme, randomly choosing elements for the sample set requires an unnecessarily large number of samples to ensure stability. As an alternative, we investigate an importance sampling strategy that utilizes the shape of the solution for loaded elastic materials to guide the sampling process.

In importance-based sampling, samples are chosen based on a probability function $p(x)$, which is similar in shape to the function being integrated. For our application, we do not know the exact shape of $\tilde{\mathbf{F}}$, but we do know that contributions from the basis functions decrease rougly cubicly as a function of distance from the surface load (Eqn. (4)), and thus $1/\rho^3$ can give a better probability distribution than uniform sampling.

Since we will sample at most Cr regions (§4.2), we limit the number of sampled elements to r per region. This gives a complexity of $O(r^3)$ for evaluating Eqns. (11)-(13). During precomputation, we sample r points in $p(x)$ and insert them into a uniform grid [Teschner et al. 2003]. For each sample point, we generate three random numbers: two uniformly random numbers between 0 and $\pi/2$ and one number between 0 and κ_o with an inverse cubic power distribution. This samples one octant of a sphere with radius κ_o. These sample points are duplicated for the remaining octants, to prevent a sampling bias in any particular octant.

During runtime sampling, we traverse the mesh elements away from the center of the region, translate them to the origin, and query against the precomputed grid to see if they contain a sample point. This traversal ends when we have r samples or we have reached the radius κ_o. This process is $O(r^2)$, since querying the grid is a constant operation.

Computing weights. With the key element set $\mathcal{C}$ selected, we still need the weights w_c to compute Eqn. (15). In our experiments, we found that error in the reduced force computation were most likely to affect stability, followed by mass, with the stiffness matrix the most resilient to error. For this reason we have focused our sampling and weighting strategy to minimize error in the forces, and re-used the weights for other computations.

In contrast to An et al. [2008], we treat this as a local problem in the sense that each weight is solved for individually, by minimizing $|\mathcal{C}|$ equations of the form

$$\sum_{j \in \mathcal{E}_i} \mathbf{w}_j = w_i \mathbf{w}_i,$$

where $\mathcal{E}_i$ is the subset of elements whose centroids are closest to the element i in the set $\mathcal{C}$ (Fig. 7), and $\mathbf{w}$ are the $3 \times r'$ basis entries for the element centroid flattened into a single $3r'$ vector. This equation can be solved directly for w_i, giving a weighting that attempts to best approximate the basis function magnitudes for all elements that this representative element substitutes. Intuitively, this approach chooses the best weight that, on average, represents the basis contribution of each element in the set.

Our results show this approach is a good compromise for online computation speed and relative accuracy. Fig. 8 shows relative error in the reduced force vector for different size samples versus a random sampling strategy (within the κ_o sphere). For this test we used a 3d version of Fig. 5, i.e., a sphere pressed into a gelatinous block. We also tested greedily using the r closest elements to the region center, but surprisingly this performed worse than the random strategy.

As a rough heuristic, our measurements revealed that stability is maintained when the relative error in reduced forces is less than 1. Future work could investigate adaptive strategies intended to maintain relative error below such a threshold. In our examples we adjust sampling so that $\mathcal{C}$ contains $O(r^2)$ elements, even when fewer may suffice.

6 Modified algorithm

Alg. 1 shows the subspace integration algorithm modified to support local basis vectors, and the following subsections describe the functions needed to update vectors and compute reduced quantities. We used one step of Backward Euler for integration in all our examples, although any method could be used. We solve the linear system of equations in Line 6 using Conjugate Gradient. For additional details, we include the source code for the full reference implementation used to generate all examples.

Algorithm 1 Use Backward Euler integration to step subspace positions and and velocities by time $h = t^{n+1} - t^n$

1: STEP($\mathbf{q}^n, \dot{\mathbf{q}}^n, h$)
2: $\mathbf{q} = \mathbf{q}^n + h\dot{\mathbf{q}}^n$
3: $(\tilde{\mathbf{F}}_{\text{ext}}, \mathcal{L}) = \text{getSurfaceLoads}(\mathbf{q})$ {§6.1}
4: $\mathbf{U} = \text{updateBasis}(\mathcal{L})$ {§6.2}
5: $(\tilde{\mathbf{F}}_{\text{int}}, \tilde{\mathbf{K}}, \tilde{\mathbf{M}}) = \text{computeReducedQuantities}(\mathbf{q}, \mathbf{U}, \mathcal{L})$ {§6.3}
6: $\Delta\dot{\mathbf{q}}^{n+1} = -(\tilde{\mathbf{M}} + h^2\tilde{\mathbf{K}})^{-1}(\tilde{\mathbf{F}}_{\text{int}} + \tilde{\mathbf{F}}_{\text{ext}})h$
7: $\dot{\mathbf{q}}^{n+1} = \dot{\mathbf{q}}^n + \Delta\dot{\mathbf{q}}^{n+1}$
8: $\mathbf{q}^{n+1} = \mathbf{q}^n + \dot{\mathbf{q}}^{n+1}$

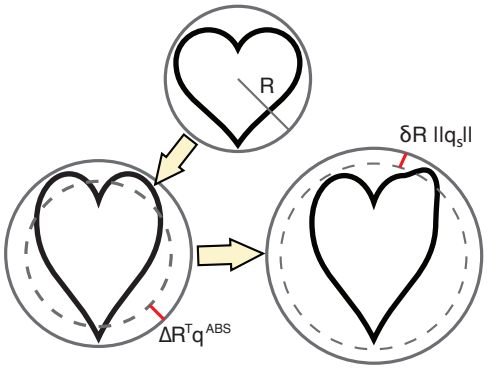


Figure 9: *In addition to the change in sphere radius introduced by global deformations, we add an additional, small change due to local displacements without degrading bound quality.*

6.1 Surface loads

Algorithm 2 Get the reduced external forces and the set of loaded vertices $\mathcal{L}$.

1: GETSURFACELOADS($\mathbf{q}$)
2: $(\mathbf{F}_{\text{contact}}, \mathcal{L}_{\text{contact}}) = \text{getCollisionLoads}(\mathbf{q})$
3: $(\mathbf{F}_{\text{other}}, \mathcal{L}_{\text{other}}) = \text{getOtherLoads}(\mathbf{q})$
4: $\mathcal{L} = \mathcal{L}_{\text{contact}} \cup \mathcal{L}_{\text{other}}$
5: **for** $l \in \mathcal{L}$ **do**
6: $\quad \tilde{\mathbf{F}}_{\text{ext}} = \mathbf{U}_l^T(\mathbf{F}^l_{\text{contact}} + \mathbf{F}^l_{\text{other}})$
7: **end for**
8: **return** $(\tilde{\mathbf{F}}_{\text{ext}}, \mathcal{L})$

In our examples, surface loads come in the form of collisions and contact, although any source could be used, e.g., interactive tools. We compute the forces present and store the set of full-coordinate vertices at which they are applied. As new local forces are applied throughout the simulation, we sparsely project these into the subspace using the respective blocks of $\mathbf{U}$ (Line 5). $\mathbf{U}_l$ is the $3 \times r'$ block of $\mathbf{U}$ corresponding to vertex l. $\mathbf{F}^l$ denotes the 3-vector block of a $3n$ vector $\mathbf{F}$.

Collisions and contact. In the special case of collisions and contact, it is possible to find the set $\mathcal{L}$ in sub-linear time. James and Pai [2004] obtain reduced collision times using a bounded deformation, or BD-Tree. Bounding volumes in this tree are efficiently updated using only subspace coordinates $\mathbf{q}$. While we cannot precompute the necessary values, we can do an efficient update that accounts for any (small) change in radius due to deformations in our local basis.

Following §3.5 from the aforementioned paper, as the local basis $\mathbf{W}$ is updated, we compute the scalar

$$\delta R = \max_l \|\mathbf{W}_{l:}\|$$

where $\mathbf{W}_{l:}$ are the s-vectors of the rows corresponding to vertices $l \in \mathcal{L}$. Then, when updating the bounding sphere radius, an additional change $\delta R\|\mathbf{q}_s\|$ is added, where $\mathbf{q}_s$ is the s-vector representing displacements in the local basis. Please consult the original paper for full BD-tree implementation details.

In practice, our experiments show that this additional change is quite small, comprising less than a 5% change in sphere radius, and causes essentially no additional false positives. Fig. 9 shows a simple example of this change in radius

Algorithm 3 Update the basis $\mathbf{W} \subset \mathbf{U}$ based on the active vertex set $\mathcal{L}$.

1: UPDATEBASIS($\mathcal{L}$)
2: $\tilde{\mathbf{U}} = \mathbf{U}$
3: $\mathcal{R} = \text{getDisjointSets}(\mathcal{L})$
4: **for** $r \in \mathcal{R}$ **do**
5: $\quad \mathcal{V} = \text{clusterSubRegions}(r, r * A_i/A)$
6: $\quad$ **for** $v \in \mathcal{V}$ **do**
7: $\quad\quad \mathbf{B}_r^x \stackrel{+}{=} \mathbf{b}_{\text{analytic}}(\mathbf{F}_v^x)$
8: $\quad\quad \mathbf{B}_r^y \stackrel{+}{=} \mathbf{b}_{\text{analytic}}(\mathbf{F}_v^y)$
9: $\quad\quad \mathbf{B}_r^z \stackrel{+}{=} \mathbf{b}_{\text{analytic}}(\mathbf{F}_v^z)$
10: $\quad$ **end for**
11: $\quad \mathbf{U} = \mathbf{U} \cup (\mathbf{B}_r^x \ \mathbf{B}_r^y \ \mathbf{B}_r^z)$
12: **end for**
13: $\mathbf{q} = \tilde{\mathbf{U}}^T\mathbf{U}\mathbf{q}$
14: $\dot{\mathbf{q}} = \tilde{\mathbf{U}}^T\mathbf{U}\dot{\mathbf{q}}$
15: **return** $\mathbf{U}$

6.2 Basis updates

This function describes how to update basis functions based on the set $\mathcal{L}$ of vertices actively deformed via a dynamic local basis.

In Line 3 we split $\mathcal{L}$ up into disjoint sets based on 1-ring connectivity. This defines the loaded regions. Then, for each region we cluster the nodes further based on how many basis vectors we can allocate to this region, based on relative surface area (A_i/A).

In our implementation, basis vectors at a node v are cached, then we check the cache to see if this node's vector has already been computed before proceeding with Lines 7–9.

6.3 Computing reduced quantities

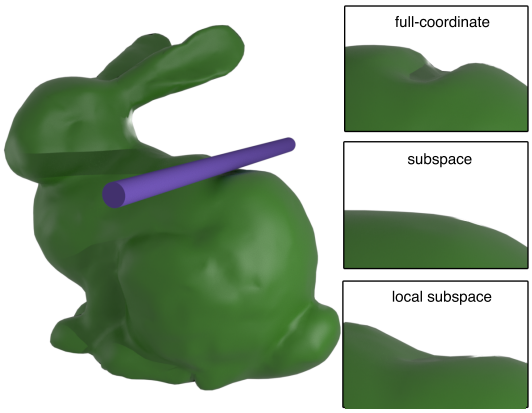
Algorithm 4 Compute the subspace quantities necessary to perform time integration.

1: COMPUTEREDUCEDQUANTITIES($\mathbf{q}, \mathbf{U}, \mathcal{L}$)
2: $(\tilde{F}_r, \tilde{\mathbf{K}}) = \text{evaluatePolynomials}(\mathbf{q})$
3: $(\mathcal{C}, \mathbf{w}) = \text{sampleElements}(\mathbf{U}, \mathcal{L})$ {§5.2}
4: **for** $e \in \mathcal{C}$ **do**
5: $\quad \tilde{\mathbf{K}}_{\text{tr}} \stackrel{+}{=} w_e\mathbf{V}_e^T\mathbf{K}_e\mathbf{W}_e$
6: $\quad \tilde{\mathbf{K}}_{\text{br}} \stackrel{+}{=} w_e\mathbf{W}_e^T\mathbf{K}_e\mathbf{W}_e$
7: $\quad \tilde{\mathbf{M}}_{\text{tr}} \stackrel{+}{=} w_e\mathbf{V}_e^T\mathbf{M}_e\mathbf{W}_e$
8: $\quad \tilde{\mathbf{M}}_{\text{br}} \stackrel{+}{=} w_e\mathbf{W}_e^T\mathbf{M}_e\mathbf{W}_e$
9: $\quad \tilde{\mathbf{F}}_s \stackrel{+}{=} w_e\mathbf{W}_e^T\mathbf{F}_e$
10: **end for**
11: **return** $(\tilde{\mathbf{F}}, \tilde{\mathbf{K}}, \tilde{\mathbf{M}})$

In Alg. 4, the function evaluatePolynomials uses precomputed data [Barbič and James 2005] or cubature optimization [An et al. 2008] to evaluate the first r sub-vector of $\tilde{\mathbf{F}}$ and the (r, r) top-left sub-block of $\tilde{\mathbf{K}}$. The subscripts tr and br denote the (r, s) top-right and (s, s) bottom-right sub-blocks of the respective matrices.

7 Results

Our results demonstrate displacement due to local basis vectors in a variety of commonplace situations. While not indistinguishable from full-coordinate motion, the resulting dynamics are plausible and achieve a reasonable trade-off between visual quality and runtime speed. See Table 1 for full simulation parameters and timings, and the accompanying video for animations.



Bunny. In this example of a cylinder repeatedly pressing into a bunny's back, we see a noticeable difference, not just in the locally deformed region, but also in the entire dynamics of the bunny motion. The local subspace method affects the global behavior by absorbing energy that would have induced global displacement.

Fig. 10 shows the timing information plotted over simulation time. We see that computing the reduced quantities dominates runtime, and cost of updating basis functions is negligible, especially as it is amortized over the entire simulation costs. The algorithm is output-sensitive, so as external loads are removed and local vibrations settle, runtime returns to that of ordinary subspace simulation. About half of the time for reduced quantities is assembling the top- and bottom-right stiffness matrix (Eqn. (11)), about 10% is computing key element force and stiffnesses (they are computed simultaneously), and about 35% in assembling the global basis components.

Falling block. Local displacements add visual detail not possible with global subspace methods. In some cases, such as the bumpy block in Fig. 2, it can significantly affect the entire motion of the object. Timings for this example only include from the moment the block hits the cylinders until the last local mode dissipates.

With global subspace methods, the small bumps on the block exterior cannot deform without introducing a significant number of basis vectors. As a result, the block cannot pass through the two unfortunately-spaced cylinders. By adding local basis vectors specific to the forces induced by the cylinders, our method allows the block to pass in the same manner as the full-coordinate simulation.

Cheburashka. In Fig. 1 we show a mesh that has been repeatedly fired upon by projectiles. The mesh exhibits plastic deformation in the local basis vector coordinates. We use a non-physical form of plasticity, and simply transfer a small fraction $(1/100)$ of $\Delta\mathbf{q}$ into the rest position each timestep. This is a simple extension that expands the capabilities of the presented method.

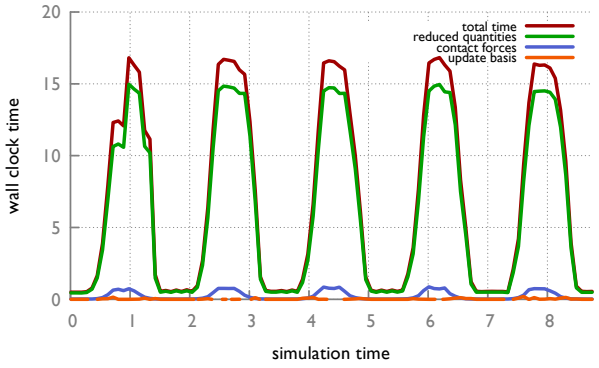


Figure 10: *The total cost per frame as well as the cost for computing reduced quantities, contact forces, and updating basis functions during the bunny simulation. This method is output-sensitive, reverting to global subspace runtimes in the absense of local deformations.*

8 Discussion

We have presented a straightforward addition to existing subspace methods that allows the representation of truly local deformations with a minimal addition of basis functions, increasing the expressive ability of subspace simulations. We also investigated the associated problems with a dynamic basis, and presented a cubature scheme that allows local dynamics with alleviated runtimes.

The primary limitation of this work is in the size of the available displacements The superposition of our linear modes can approximate some non-linear behavior, but as the deformation region approaches the size of the object, computational benefits can quickly be diminished as an excessive number of samples are required for stable simulation. The result is a method that works very well for a global basis combined with very local modes, but poorly for the intermediate regime.

One secondary limitation of local displacements is the restriction on timestep size. While a few global modes can be simulated at large timesteps, adding local modes imposes a limit in order to represent the physics of the system. This is unavoidable, and the user should bear in mind the limits imposed by the CFL condition for the dynamics of interest.

Future work. Barbič and James [2010] precompute possible deformations of modes that cause (self-)collisions, allowing for incredibly fast self-collision detection. We do not explore local dispacements due to self-collisions in this paper, but since deformations due to our local bases are quite small, an extension to this work would be straight-forward, for example by following a similar treatment to that in §6. Likewise, augmenting the work of Kim and James [2011] and Barbič and Zhao [2011] with local displacements would provide the user with a full-range of deformations, from truly local, through intermediate and up to global deformations.

Formulating a theoretical basis for cubature sampling remains an open problem with many practical benefits, since any improvements in sampling yield runtime improvements, both in stability and simulation time. Furthermore, a cubature theory would unify sampling for arbitrary basis functions, such as those presented here and in An et al. [2008], a well as opening the door for dynamically computing basis functions of arbitrary physical phenomena.

Scene	Vertices	Tets	E GPa	ν	ρ (kg/m^3)	FPS (1/sec)	Δt (sec)	r	Full (sec)	Subspace (sec)	Local (sec)	Speedup
Bunny	8372	44757	0.05	0.25	1.0	120	2.5×10^{-4}	28	219.3	0.57	1.60	137x
Bumpy	13156	45495	0.5	0.35	0.92	720	10^{-6}	16	168.5	0.166	2.25	75x
Cheburashka	11989	43813	0.005	0.25	1.0	300	2×10^{-4}	32	111.6	0.007	0.183	610x

Table 1: *All examples were run as a single thread on a 3.6GHz i5 with 12GB RAM. Simulation times are average times per frame.*

Acknowledgements

This work was supported by the NSF (DMS-0602235, IIS-0905502, OCI-1047932). The Mitsuba renderer was used for all examples.

References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5, 165:1–165:10.

BARBIČ, J., AND JAMES, D. 2005. Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 982–990.

BARBIČ, J., AND JAMES, D. 2010. Subspace self-collision culling. *ACM Transactions on Graphics (TOG) 29*, 4, 81.

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

BATHE, K.-J., AND GRACEWSKI, S. 1981. On nonlinear dynamic analysis using substructuring and mode superposition. *Computers & Structures 13*, 5—6, 699 – 707.

DEBUNNE, G., DESBRUN, M., CANI, M., AND BARR, A. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 31–36.

FAURE, F., GILLES, B., BOUSQUET, G., AND PAI, D. K. 2011. Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4 (July), 73:1–73:10.

GERSZEWSKI, D., BHATTACHARYA, H., AND BARGTEIL, A. W. 2009. A point-based method for animating elastoplastic solids. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. Charms: a simple framework for adaptive simulation. In *ACM Transactions on Graphics (TOG)*, vol. 21, ACM, 281–290.

HAUTH, M., AND STRASSER, W. 2004. Corotational simulation of deformable solids. In *Journal of WSCG*, 137–145.

IDELSOHN, S. R., AND CARDONA, A. 1985. A load-dependent basis for reduced nonlinear structural dynamics. *Computers & Structures 20*, 1—3, 203 – 210.

IDELSOHN, S. R., AND CARDONA, A. 1985. A reduction method for nonlinear structural dynamic analysis. *Computer Methods in Applied Mechanics and Engineering 49*, 3, 253 – 279.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 65–72.

JAMES, D. L., AND PAI, D. K. 2003. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph. 22*, 1 (Jan.), 47–82.

JAMES, D. L., AND PAI, D. K. 2004. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph. 23*, 3 (Aug.), 393–398.

JOHNSON, K. 1987. *Contact mechanics*. Cambridge Univ Pr.

KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. 28*, 5, 123:1–123:9.

KIM, T., AND JAMES, D. L. 2011. Physics-based character skinning using multi-domain subspace deformations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '11, 63–72.

KOYAMA, Y., TAKAYAMA, K., UMETANI, N., AND IGARASHI, T. 2012. Real-time example-based elastic deformation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 19–24.

M., O., Y., L., AND A., C. 1996. Hybrid simulation strategy for multiple planar collisions with changing topologies and local deformation. *Finite Elements in Analysis and Design 23*, 2, 225–239.

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-Based Elastic Materials. *SIGGRAPH (ACM Transactions on Graphics) 30*, 4 (Aug), 72:1–72:8.

NICKELL, R. 1976. Nonlinear dynamics by mode superposition. *Computer Methods in Appl. Mech. and Eng. 7*, 1, 107 – 129.

PAULY, M., PAI, D. K., AND GUIBAS, L. J. 2004. Quasi-rigid objects in contact. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '04, 109–119.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: modal dynamics for graphics and animation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM, SIGGRAPH '89, 215–222.

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Proc. VMV*, 47–54.

THOMSON, W. 2004. *Theory of vibration with applications*. Taylor & Francis.

WU, X., DOWNES, M., GOKTEKIN, T., AND TENDICK, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Computer Graphics Forum*, vol. 20, Wiley Online Library, 349–358.