Multigrid Integration for Interactive Deformable Body Simulation

Xunlei Wu[1] and Frank Tendick[2]

[1] Simulation Group, CIMIT/Harvard University,
65 Landsdowne St., Cambridge, MA 02139, USA
wu.xunlei@mgh.harvard.edu
[2] Department of Surgery, University of California San Francisco,
513 Parnassus Ave., Room S-550, San Francisco, CA 94143, USA
tendick@eecs.berkeley.edu

Abstract. Simulation of soft tissue behavior for surgical training systems is a particularly demanding application of deformable modeling. Explicit integration methods on single mesh require small time step to maintain stability, but this produces slow convergence spatially through the object. In this paper, we propose a multigrid integration scheme to improve the stability and convergence of explicit integration. Our multigrid method uses multiple unstructured independent meshes on the same object. It is shown that, with the proposed multigrid integration, both stability and convergence can be improved significantly over single level explicit integration.

1 Introduction

Simulation of the behavior of soft biological tissue for surgical training or planning systems is a particularly demanding application of deformable modeling. Tissue is highly nonlinear, commonly showing an exponential stress-strain relationship [1]. Large deformations of 100% or more are possible [1]. Yet, simulation requires realistic behavior in real time. High accuracy at interactive speeds is necessary for intra-operative surgical planning (e.g., [2]).

Recent research in soft tissue modeling has focused on finite element methods (FEM) for accuracy. Initial efforts (e.g., [3,4,5]) used linear models because of their speed and because they permit significant offline pre-computation to reduce runtime computation. Linear methods cannot handle large deformation or material nonlinearity, however. Cotin et al. approximated nonlinear elasticity within an otherwise linear model [6]. Picinbono et al. [7] and Müller et al. [8] extended linear methods to account for rotation that occurs with large deformation.

In nonlinear FEM, strain is nonlinear in displacement (geometric nonlinearity) while stress is nonlinear in strain (material nonlinearity). Szekely et al. [9] used a large scale multi-processor computer to obtain real-time performance while modeling both types of nonlinearity. Zhuang and Canny [10] modeled geometric nonlinearity for 3-D deformable objects, but not nonlinear material properties. Zhuang used mass lumping to produce a diagonal mass matrix for speed.

Wu et al. [11] also used mass lumping, but incorporated both types of nonlinearity. In this paper, we use the same modeling engine as [11].

Each of these nonlinear implementations uses explicit time integration for real-time response. In the animation community, implicit integration methods have become popular, spurred in part by the work of Baraff and Witkin [12]. While implicit methods permit large time steps in animation, convergence is typically too slow for real-time simulation of 15 or more frames per second if nonlinear behavior is accommodated. However, two recent papers have used implicit integration with finite element models for simulation. Müller et al. [8] warp a precomputed linear stiffness matrix according to the local rotation of the material to account for large deformations. However, their method only permits approximation of nonlinear stress-strain relationships, and cannot accommodate a general strain energy function. Capell et al. [13] use a multiresolution subdivision framework that permits real time deformation, but this does not have the accuracy or flexibility of, e.g., tetrahedral meshes.

Explicit integration schemes use much less computation per integration step, but require a small step size to maintain stability. The critical step size is proportional to element size and inversely proportional to the square root of stiffness [14]. Although the softness of deformable objects makes explicit integration feasible, there is a tradeoff between spatial resolution, i.e. the fineness of the grid, and the material stiffness that can be achieved.

The other major disadvantage of explicit integration of dynamic models is that information is propagated spatially at only one layer per time step. Although the user's motion while interacting with a surgical simulation is likely to be slow, the "slinky"-like motion that results from slow propagation is visually unrealistic. The stress concentration that results from local perturbations can also cause instability before the stress is distributed by the integrator.

In this paper, we take a multiresolution approach to integration. Although there has been significant effort in multiresolution approaches in deformable modeling (e.g., [11,15,16]), past work has emphasized adaptation for providing local detail. Instead, our integration scheme maintains simultaneous multiple resolutions of a triangle or tetrahedral mesh of the same object. The scheme is an extension of traditional multigrid methods in which stress due to a local perturbation is rapidly distributed through an object by restriction onto coarser mesh representations. After integration on the coarser meshes, results are projected back to finer meshes for refinement. The result is significantly faster convergence and better stability than single level explicit integration.

1.1 Multigrid Iterative Solver Background

Multigrid (MG) methods are also called multilevel, multi-scale, aggregation, and defect correction methods[17]. In the beginning, they were invented as one kind of iterative scheme to solve for X in

$$T(X) = b \tag{1}$$

where T can be either a linear or nonlinear operator mapping the current state X to internal stress, particularly for partial differential equations (PDE). In contrast to single level iterative solvers, MG uses coarse grids to do divide-and-conquer in two related senses. First, it obtains an initial solution from the fine grid by using a coarse mesh as an approximation. This is done by transforming the problem on the fine grid to the coarse grid and then improving it. The coarse grid is in turn approximated by a coarser grid, and so on recursively. The second way MG uses divide-and-conquer is in the frequency domain. This requires us to think of the error as a sum of eigenvectors of different frequencies. Then, intuitively, the work on a particular grid will attenuate the high frequency error by averaging the solution at each vertex with its neighbors. There are three operators used in the conventional MG iterative solver [18]:

- The smoothing operator $\mathbf{G}(\cdot)$ takes a problem and its approximated solution $X^{(i)}$ and computes an improved $X^{(i)}$ using a one-level iterative solver such as Gauss-Seidel or Jacobi methods. This smoothing is performed on all but the coarsest mesh, where an exact solution of (1) is formed:

$$X^{(i)} = \begin{cases} \mathbf{G}(X^{(i)}, b^{(i)}), \, i \neq (m-1) \\ T^{-1}(b^{(m-1)}), \, i = (m-1) \end{cases} \tag{2}$$

  The superscript (i) indicates $\mathbf{G}(\cdot)$ operates on grid level i, for m mesh levels numbered from 0 (finest) to $m-1$ (coarsest).
- The restriction operator $\mathbf{R}(\cdot)$ takes the residual of (1), $T(X^{(i)}) - b^{(i)}$, and maps it to $b^{(i+1)}$ on coarser level $i+1$.

$$b^{(i+1)} = \mathbf{R}(T(X^{(i)}) - b^{(i)}) \tag{3}$$

- The interpolation operator $\mathbf{P}(\cdot)$ projects the correction to an approximate solution $X^{(i)}$ on the next finer mesh.

$$X^{(i)} = X^{(i)} - \mathbf{P}(X^{(i+1)}) \tag{4}$$

The three operators manage a series of grids with different resolutions and form a "V" shape algorithm as shown in Figure 1.

1.2 Overview

In this paper, we extend traditional multigrid methods to apply them to explicit integration of dynamic models of the form

$$M\ddot{U} + D\dot{U} + R(U) = V \tag{5}$$

where, for a finite element model, M and D are mass and damping matrices, respectively, U is the nodal displacement vector, R is the internal stress operator, and V is the traction force vector. The methods operate on unstructured independent triangle or tetrahedron meshes. We developed and evaluated four variations, each using a single level explicit integrator in place of the smoother $\mathbf{G}(\cdot)$ of traditional multigrid methods [19]. Different restriction and projection operators were used, producing different stability and convergence properties. Here we present the single method which combined the best properties of the four.

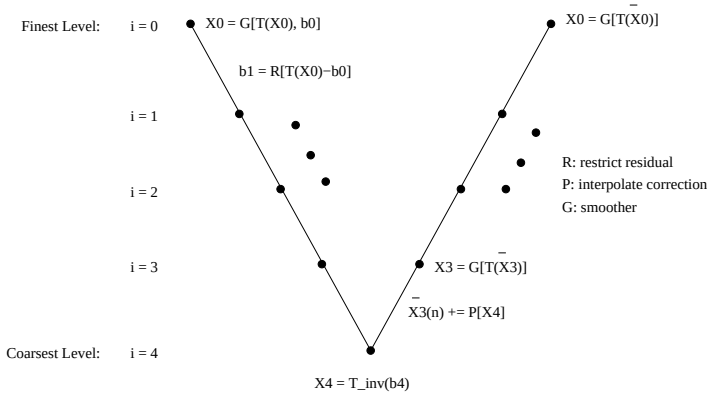

Fig. 1. A V-cycle of traditional multigrid iterative solver solving equation (1).

2 Methodology

In the current implementation, we use the FEM engine of Wu et al. [11]. Details can be found in that reference. The FEM model uses a total Lagrangian formulation to accommodate geometric nonlinearity and multiple strain energy density functions have been implemented to represent the hyperelastic behavior of soft tissue. Mass lumping and Rayleigh damping are used to create a diagonal mass matrix M and associated damping matrix D. Consequently, the method scales computationally as $O(n)$, where n is the number of nodes in the 2D or 3D mesh.

2.1 Mesh Correspondence

Our multigrid method uses unstructured triangle or tetrahedron meshes. The meshes are independent, i.e., no vertex needs to be shared between levels. The m levels of mesh, $\mathbf{S^{(i)}}$ for $i = 0, m-1$, are discretized versions of the continuous domain $\mathbf{S}$. They can be chosen so that the numbers of vertices and elements in the levels form a geometric series with a predefined ratio. Because the FEM engine has linear computational cost $O(n)$, the geometric series is the key to achieving $O(n_0)$ cost as will be shown in section 2.3, where n_0 is the number of vertices in the finest mesh $\mathbf{S^{(0)}}$. Off-the-shelf 2D and 3D mesh generators, such as Triangle [20] or Gmsh [21], can be used to triangulate or tessellate deformable objects, respectively. Both programs can control the output mesh density by setting a resolution parameter.

The vertices and elements in each $\mathbf{S^{(i)}}$ are mapped to other levels through an initial mapping process. Each vertex on level i has host elements on levels $i-1$ and $i+1$, for $i \in [1, m-2]$. Each node in $\mathbf{S^{(0)}}$ and $\mathbf{S^{(m-1)}}$ only has host elements in $\mathbf{S^{(1)}}$ and $\mathbf{S^{(m-2)}}$, respectively. The correspondence is invariant throughout the simulation. As shown in the left of Figure 2, the host element of

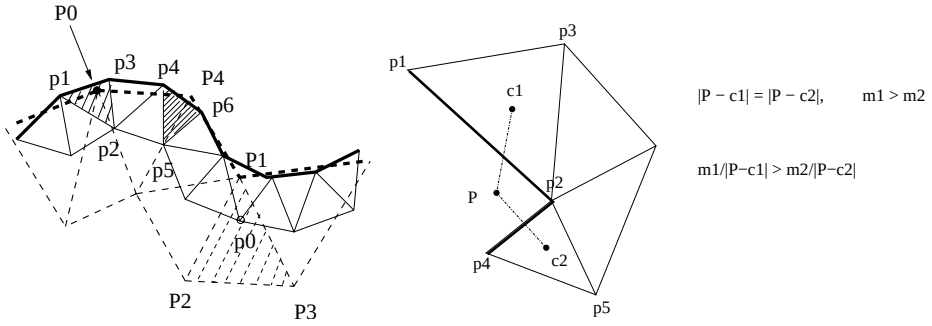

Fig. 2. *Left:* **Mesh correspondence in 2D**. P_i and p_i are vertices in coarse mesh $\mathbf{S^{(i-1)}}$ and fine mesh $\mathbf{S^{(i)}}$, respectively. The thick solid line and the thick dashed line indicate the boundaries of domains $\mathbf{S^{(i)}}$ and $\mathbf{S^{(i-1)}}$, respectively. The fine face (p_1, p_2, p_3) is identified as the host element of coarse vertex P_0. Fine vertex p_0 is associated with coarse face (P_1, P_2, P_3). P_4 is located outside of the contour of fine mesh. Its host element is identified as face (p_4, p_5, p_6), because it is the closest triangle to P_4 measured by (7). *Right:* **Mapping boundary vertices**. c_1 and c_2 are the centroids of faces (p_1, p_2, p_3) and (p_2, p_4, p_5), which are boundary elements. P is outside of the discretized domain. Although P has the same distance to c_1 and c_2, i.e. $\gamma_{P,1} = \gamma_{P,2}$, P's host element is classified as face (p_1, p_2, p_3) because $\frac{m_1}{\gamma_{P,1}} > \frac{m_2}{\gamma_{P,2}}$.

vertex P_0 is identified as face (p_1, p_2, p_3), which encloses P_0. The same is true for face (P_1, P_2, P_3), which hosts vertex p_0. The weights for each coarse node are stored in a vector ω of size $npe \times 1$, where npe stands for *number of nodes per element*. All interior nodes of coarse mesh $\mathbf{S^{(i)}}$ lie in the convex hull formed by the fine mesh $\mathbf{S^{(i-1)}}$, and vice versa. Thus the associated weights for each coarse node ω_j are non-negative and sum to 1. ω_j is computed as

$$\omega_j \equiv \begin{cases} \frac{A_j}{A}, \text{ Triangle mesh} \\ \\ \frac{V_j}{V}, \text{ Tetrahedron mesh} \end{cases}$$

where A_j is the area of the sub-triangle involving the test point and two points from the host triangle. V_j is the volume of the sub-tetrahedron involving the test point and three points from the host pyramid. Thus the location of node $p_\theta^{(i)}$ can be represented as a weighted sum of the positions of corner nodes of its hosting element γ on $\mathbf{S^{(i-1)}}$ or those of its hosting element η on $\mathbf{S^{(i+1)}}$,

$$p_\theta^{(i)} = \begin{cases} \sum_{j=1}^{npe} \omega_{\gamma,j}^{(i-1)} p_{\gamma,j}^{(i-1)}, \; ele_\gamma \in \mathbf{S^{(i-1)}} \\ \\ \sum_{j=1}^{npe} \omega_{\eta,j}^{(i+1)} p_{\eta,j}^{(i+1)}, \; ele_\eta \in \mathbf{S^{(i+1)}} \end{cases} \tag{6}$$

where p is the position vector of a node.

Only boundary nodes of the coarse mesh are allowed to have negative weights or weights that are larger than 1. An element ele_i is the host element of the boundary node θ on another mesh if ele_i encloses θ. If θ is outside all the elements on another mesh, then θ is mapped to the boundary element ele_i on another level which maximizes

$$\frac{m_i}{\gamma_{\theta,i}} = \max_k \frac{m_k}{\gamma_{\theta,k}} \tag{7}$$
$$\gamma_{\theta,k} \equiv \|p_\theta - \frac{1}{npe}\sum_{l=1}^{npe} p_{k,l}\|.$$

$\gamma_{\theta,k}$ is the Euclidean distance between θ and the centroid of element k. As shown in the right half of Figure 2, if a vertex P is equidistant to the centroids of faces (p_1, p_2, p_3) and (p_2, p_4, p_5), face (p_1, p_2, p_3) is identified as the host element if this face has a larger area.

Each coarse vertex associates with a hosting element in the finer mesh. It is important that the hosting elements for the vertices in $\mathbf{S^{(i)}}$ cover $\mathbf{S^{(i-1)}}$. This guarantees that the state of every fine vertex can be represented in $\mathbf{S^{(i)}}$ so that detail will not be lost during the restriction process $\mathbf{R}(\cdot)$. This constrains the resolution ratio r between the number of vertices in $\mathbf{S^{(i)}}$ and $\mathbf{S^{(i-1)}}$. In practice, we have found $r = 0.5$ to be an effective ratio.

Franklin's *pnpoly* is used to test whether a point is inside of a triangle. The test of a point being inside a tetrahedron is conducted by checking if all four sub-tetrahedra have positive volumes provided that the tested tetrahedron is not degenerate. After the correspondence between vertices and elements on different levels is constructed, the ω_j's are used as the weights when restricting or interpolating quantities between mesh levels.

2.2 Multigrid Time Integrator

The proposed Multigrid time integrator is called Restrict-and-Interpolate-State-or-Variation (RaISoV). Similar to the MG iterative solver, RaISoV uses a restriction operator $\mathbf{R^i_{i-1}}(\cdot)$ in the down stroke of the V-cycle in Figure 1 and an interpolation operator $\mathbf{P^i_{i-1}}(\cdot)$ in the up stroke to transfer the problem from one level to another. Also in the up stroke, it applies a solution operator $\mathbf{G}(\cdot)$ on each level mesh $\mathbf{S^{(i)}}$.

Restriction operator At current time step n, $\mathbf{R^i_{i-1}}(\cdot)$ assigns either the state of each node $X^{(i-1)}$ or the variation of nodal state $\Delta X^{(i-1)}$

$$\Delta X^{(i-1)} \equiv X_n^{(i-1)} - X_{n-1}^{(i-1)} \tag{8}$$

in $\mathbf{S^{(i-1)}}$ to that in $\mathbf{S^{(i)}}$ in a weighted sum fashion.

$$X_\Theta^{(i)} = \begin{cases} \mathbf{R^i_{i-1}}(X_\gamma^{(i-1)}) = \sum_j \omega_{\gamma,j}^{(i-1)} X_{\gamma,j}^{(i-1)}, & |\sum_j \omega_{\gamma,j}^{(i-1)} \Delta X_{\gamma,j}^{(i-1)}| > \epsilon \\ X_\Theta^{(i)} + \mathbf{R^i_{i-1}}(\Delta X_\gamma^{(i-1)}) = X_\Theta^{(i)} + \sum_j \omega_{\gamma,j}^{(i-1)} \Delta X_{\gamma,j}^{(i-1)}, & otherwise \end{cases} \tag{9}$$

where the subscript Θ and (γ, j) denote vertex index Θ on level i and the j^{th} corner node of hosting element index γ on level $i-1$, respectively. The weight $\omega_{\gamma,j}^{(i-1)}$ is computed as in section 2.1. The switch is conducted by checking the condition

$$|\sum_j \omega_{\gamma,j}^{(i-1)} \Delta X_{\gamma,j}^{(i-1)}| > \epsilon \tag{10}$$

When (10) is true, the weighted sum of the nodal state variation in the finer mesh is large. This indicates the regional nodes in $\mathbf{S^{(i-1)}}$ are subjected to large stress, since the state differential $\dot{X} \approx \frac{\Delta X}{\Delta}$ is large. $\mathbf{R}(\cdot)$ will linearly project those states into the corresponding vertices in the coarse mesh $\mathbf{S^{(i)}}$. If (10) is false, the algorithm regards these finer nodes as converging to their steady state. By mapping the variation, RaISoV suppresses the high frequency component of the residual $R(U) - V$ of (5). This improves the convergence rate of the system. For the proof of the linear problem please refer to Demmel [18].

Interpolation operator Similarly, the interpolation operator $\mathbf{P_{i-1}^i}(\cdot)$ projects either $X^{(i)}$ or $\Delta X^{(i)}$ in $\mathbf{S^{(i)}}$ to that on level $i-1$ using weights $\omega^{(i)}$.

$$X_\theta^{(i-1)} = \begin{cases} \mathbf{P_{i-1}^i}(X_\Gamma^{(i)}) = \sum_j \omega_{\Gamma,j}^{(i)} X_{\Gamma,j}^{(i)}, & |\Delta X_\theta^{(i-1)}| > \epsilon \\ X_\theta^{(i-1)} + \mathbf{P_{i-1}^i}(\Delta X_\Gamma^{(i)}) = X_\theta^{(i-1)} + \sum_j \omega_{\Gamma,j}^{(i)} \Delta X_{\Gamma,j}^{(i)}, \textit{otherwise} \end{cases} \tag{11}$$

θ is the nodal index in $\mathbf{S^{(i-1)}}$. Γ is the hosting element index in $\mathbf{S^{(i)}}$. The switch condition is

$$|\Delta X_\theta^{(i-1)}| > \epsilon \tag{12}$$

Although the formulations of (10) and (12) are different, their goals are consistent. By measuring the state variation in the finer mesh, we either project the state to relax the finer mesh local distortion when the nodal tangent is large, i.e. (12) is true, or interpolate the state variation to suppress the local residual in $\mathbf{S^{(i-1)}}$ for faster convergence when (12) is false. For the rest of this paper, we will simplify the notation of $\mathbf{R_{i-1}^i}(\cdot)$ and $\mathbf{P_{i-1}^i}(\cdot)$ as $\mathbf{R}(\cdot)$ and $\mathbf{P}(\cdot)$, respectively.

Solution operator RaISoV applies a conventional explicit ODE solver, such as forward Euler or Runge-Kutta, as the solution operator $\mathbf{G}(\cdot)$ in each $\mathbf{S^{(i)}}$.

$$X_{n+1}^{(i)} = \mathbf{G}(X_n^{(i)}, V^{(i)}), \quad X_n^{(i)} \equiv \begin{bmatrix} U \\ \dot{U} \end{bmatrix}_n^{(i)} \tag{13}$$

where $V^{(i)}$ is the right hand side of (5) at level i. The state of each vertex is a 6×1 vector including its displacement and velocity. This is different from the conventional MG iterative solver, which uses a direct solver at the coarsest level and a Jacobi-like pre-conditioner as a smoother in the other levels. The solution operator must be stable at each level. This condition ensures that $X_{n+1}^{(i)}$

is closer to the steady state solution of (5), $X_\infty^{(i)}$, than $X_n^{(i)}$. Currently, Δ is kept the same on every level for simplicity, though different step sizes could be used for different resolutions to accommodate individual stability requirements. If a nonlinear viscosity model is considered, the internal stress will require nonlinear mapping from both the position U and the velocity $\dot{U}$ in (5), i.e. $R(U, \dot{U})$.

The pseudo-code of RaISoV algorithm in the Appendix shows the method's schematics.

2.3 Computational Cost of MG Integrator

Within each V-cycle, there are m time integrations, where m is the number of levels. The nonlinear FEM engine we use has a linear computational cost of 300 FLOPS/node, for either 2D or 3D [11]. I.e., $C(n) = 300n$ where n is the number of nodes. This is much greater than the cost of the restriction and interpolation processes in the multigrid algorithms, 2×42 FLOPS/node for tetrahedra and 2×30 FLOPS/node for triangle meshes. Thus, if the geometric series of the mesh nodes has ratio 0.5, the total cost for each V-cycle is roughly the sum of the FEM update cost at all levels, i.e.

$$\begin{aligned} C_{RaISoV}(n^{(0)}) &= \textstyle\sum_{i=0}^{m-1} C(n^{(i)}) + Cost(mapping) \\ &\leq \begin{cases} 2[C(n^{(0)}) + 84n^{(0)}] \approx 768n, \text{ 3D mesh} \\ 2[C(n^{(0)}) + 60n^{(0)}] \approx 720n, \text{ 2D mesh} \end{cases} \end{aligned} \tag{14}$$

The two sides of the above equation become equal as m approaches infinity. The increased computational cost is compensated by the stabler performance of multigrid integration, because in practice the multigrid algorithm can be integrated with a time step larger than twice that of the single level integrator. This is shown in the 2D examples in the next section and the 3D example in the accompanying video.

3 Evaluation

In this section, RaISoV is compared to single-level integration method (SLIM). The better stability and the faster convergence of RaISoV are demonstrated with two examples. In the examples, four levels of triangle meshes are used to represent the discretization of the square membrane shown in Figure 3. The meshes were generated using *Triangle* [20]. SLIM evaluates only the finest mesh, $\mathbf{S}_0$. The membrane has size 10 cm $\times$ 10 cm and all four sides fixed in space. It is modeled with a Mooney-Rivlin material [22] with constants $C_1 = 1.0$ and $C_2 = 1.0$. The damping ratio is 1.0. Both SLIM and RaISoV use an explicit fourth-order Runge-Kutta integrator to solve the system.

3.1 Perturbation Response

In the first example, with all four sides fixed in space and free of gravity, a central node of the membrane $p_{\alpha_0}^{(0)}$ is lifted instantaneously by 3.0 cm, and the node

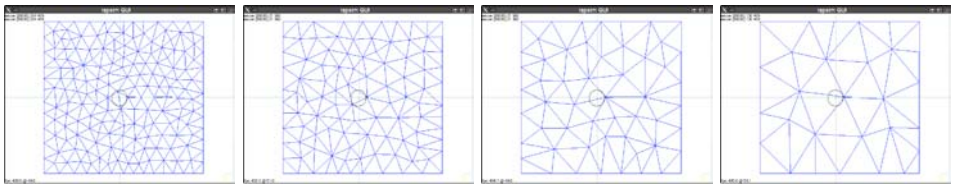

Fig. 3. Four levels of 2D triangle meshes on a square membrane: $\mathbf{S}_0$ with 186 nodes and 321 faces, $\mathbf{S}_1$ with 103 nodes and 165 faces, $\mathbf{S}_2$ with 55 nodes and 82 faces, and $\mathbf{S}_3$ with 31 nodes and 40 faces. Notice the numbers of faces at different levels form a geometric series.

stays at the height throughout the simulation (Figure 4). This test simulates the step response of the membrane with concentrated large stress surrounding the lifted node initially. The initial displacement of $p_{\alpha_0}^{(0)}$ causes the neighboring elements to be highly stretched. If a single level mesh is used to simulate this scenario, explicit time integration is going to be unstable unless a small step size $\Delta < \Delta_{crit} = \sqrt{\frac{c}{K_{max}}}$ [14] is applied, where $K_{max} = max\frac{|R(X_p)|}{|X_p|}$ is the largest eigenvalue of pseudo stiffness matrix. Note that, because of the hyper-elastic material property, K_{max} increases as an element stretches. The time evolution of $|T - T_\infty|$ of the two algorithms is plotted in the left half of Figure 5, where T is the nodal stress tensor of $\mathbf{S^{(0)}}$ caused by element deformation [11]. The stress of fixed boundary nodes is not included in the calculation. The single level algorithm is applied with time step 5 ms; RaISoV uses 10 ms as its time step.

The initial lift is passed onto the nodes $p_{\alpha_i}^{(i)}$ in coarser meshes $\mathbf{S^{(1)}}$ to $\mathbf{S^{(3)}}$ that are closest to $p_{\alpha_0}^{(0)}$. Because the concentrated stress around $p_{\alpha_0}^{(0)}$ results in large variation at those surrounding nodes, RaISoV modifies those states directly. This largely regulates the stress concentration around $p_{\alpha_0}^{(0)}$ after one V-cycle, as can be seen in the left half of Figure 5. Eliminating the stress concentration eliminates the need to reduce the time step in order to maintain stability in the event of a large perturbation. Note that in Figure 5 SLIM does not distribute the stress on coarser levels as RaISoV does, and takes far longer for stress to reach steady state. In fact, if the perturbation were increased from 3.0 to 3.5 cm, SLIM would actually be unstable unless the step size were reduced further.

After the strain distribution is smoothed, condition (10) is *not* satisfied for most nodes in the finer meshes. This "forgetting factor" eliminates the explicit mapping from the coarse meshes on successively finer levels. This improves the convergence of the result.

3.2 Draping Under Gravity

In a second example, the membrane drapes under gravity with the four sides fixed. A time step of 10 ms is used for SLIM and RaISoV. The stress distribution in this case is smooth and this scenario is used to test the convergence rates of

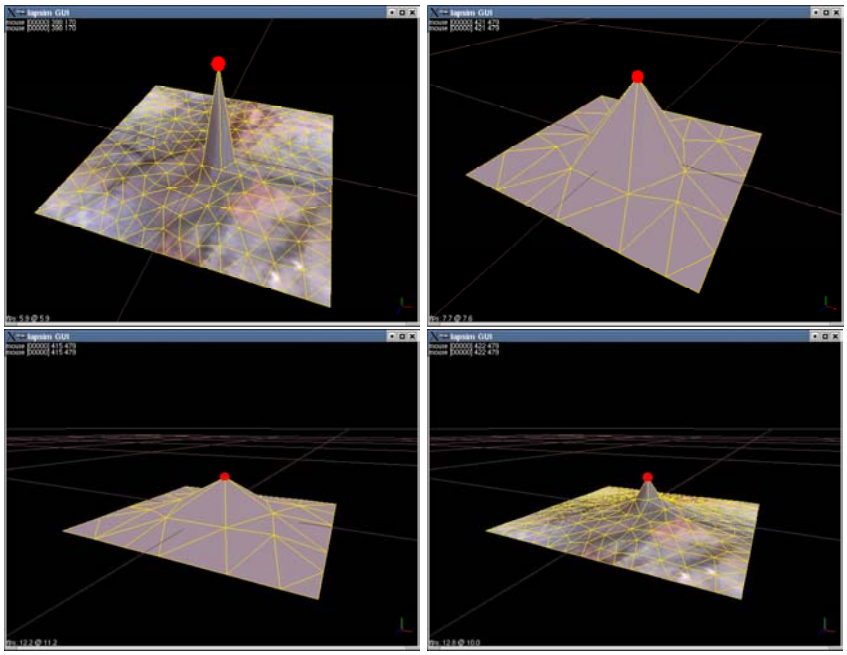

Fig. 4. *First:* One node is lifted rapidly within one time step. Neighboring elements undergo very large deformation. *Second:* The displacement is restricted onto the coarse mesh, where the distortion is distributed over a larger region by the larger elements. *Third:* As the multigrid algorithm proceeds over multiple cycles, stress redistributes through the coarse mesh. *Fourth:* Effect of coarse mesh acts to spread influence over the fine mesh, eventually reducing its effect as steady state is approached.

the two approaches. Figure 6 shows the initial and final states of the draping process.

In this experiment, the gravitational force applies evenly on every element of the mesh so that strain distribution is also smooth. Condition (10) is *not* satisfied on all nodes throughout the simulation. RaISoV converges faster than SLIM as shown in right half of Figure 5. It combines the stability benefits of distributing large strain faster and the convergence attributes of conventional MG iterative solver while maintaining a computational cost bounded by $2C(n^{(0)})$.

3.3 Element Inversion

Section 3.1 showed one way in which the multigrid approach can avoid instability by reducing the buildup of local stress. There is another way in which effective instability can occur. If an element is inverted by a perturbation large enough to cause a node to cross over the opposing edge (in a triangle element) or side (in a tetrahedron) within one time step, this will cause the model to go unstable. The

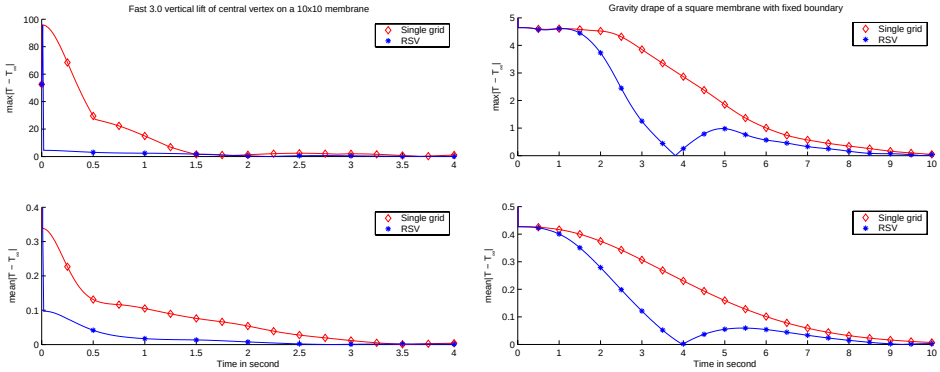

Fig. 5. *Left:* One central node of the membrane is subjected to an initial vertical lift of 3.0 cm. The results of RaISoV and SLIM are shown. *Upper left*: The time evolution of $max|T - T_\infty|$, which occurs at the lifted node $p_{\alpha_0}^{(0)}$. *Lower left*: The average of $|T - T_\infty|$ over all internal nodes. *Right:* With four sides fixed in the space, the membrane drapes under gravitational force. *Upper right*: time trajectory of $max|T - T_\infty|$. *Lower right*: $mean|T - T_\infty|$ versus time.

multigrid method reduces this possibility in two ways. First, the distribution of stress distributes the effects of displacement over multiple elements, reducing the possibility that any of them will be inverted. Second, by restricting the effect onto a coarser grid with larger elements, the size of displacement necessary to invert an element is increased. These properties make multigrid more robust to this form of instability. This is demonstrated in a 3-D example in the accompanying video.

4 Conclusion

The multigrid algorithm RaISoV is easy to implement, provides clear benefits in stability and convergence over single level explicit integration, and provides real time performance. Multiple mesh levels of triangles or tetrahedra can be generated easily offline by off-the-shelf software. As indicated in equation(14), one MG cycle requires less than twice the cost of a single level method, so MG is cheaper to execute if it can stably simulate a dynamic system with the largest permissible time step $max(h_{MG}) \geq 2max(h_{one-level})$. Because of the improved stability of MG, this is likely to be the case.

In our implementation so far, we have used the same integrator and time step at each level of the MG schemes. This is not required. Different integrators and step sizes have varying filtering properties that may produce advantages at different mesh resolutions. In fact, if the coarsest mesh has few enough nodes to permit *implicit* integration in real time, an intriguing possibility is to use *explicit* integration on the finer meshes to smooth the result while obtaining the

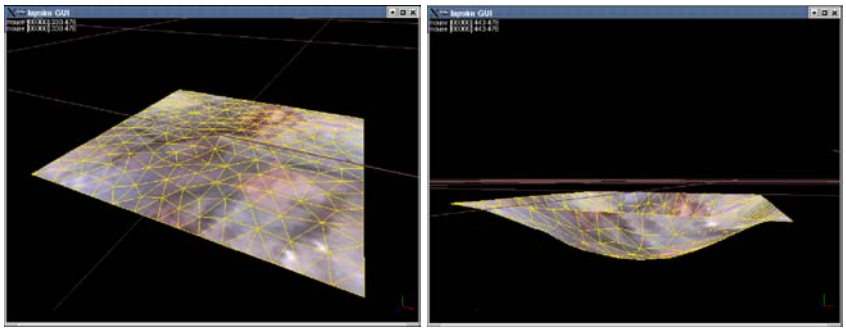

Fig. 6. With four sides fixed in the space, the membrane drapes under gravity. *Left:* Membrane before gravity is applied. *Right:* Steady state after deformation due to gravity.

stability advantages of implicit integration at the coarsest level. We are currently implementing an implicit solver for the FEM engine.

The proposed method is not suitable for the analysis of the transient response, in which the details of the evolution of the state differential $\dot{X}$ are significant. However, it is reasonable to assume that the user's movements in interacting with a surgical simulator will be relatively slow, so that dynamics can be neglected.

A significant disadvantage of the multigrid method is that the mesh levels must be generated offline in practice. This prohibits, for example, local mesh refinement when tissue is cut. However, multigrid methods could be implemented within an overall hierarchical method such as the dynamic progressive mesh scheme of Wu et al. [11]. Refinement could occur within the region of the cut. While a single level integrator would be necessary within that region, multilevel integration would still be possible elsewhere to maintain net benefits.

References

1. Fung, Y.: Biomechanics: Mechanical Properties of Living Tissues. Springer-Verlag, New York (1993)
2. Alterovitz, R., Pouliot, J., Taschereau, R., Hsu, I.C., Goldberg, K.: Simulating needle insertion and radioactive seed implantation for prostate brachytherapy. In Westwood, J., et al., eds.: Medicine Meets Virtual Reality 11, Amsterdam, IOS Press (2003) 19–25
3. Bro-Nielsen, M.: Finite element modeling in surgery simulation. Proc. IEEE **86** (1998) 490–502
4. Cotin, S., Delingette, H., Bro-Nielsen, M., Ayache, N., et al.: Geometric and physical representations for a simulator of hepatic surgery. In Weghorst, S., et al., eds.: Medicine Meets Virtual Reality: 4, Amsterdam, IOS Press (1996) 139–151
5. James, D., Pai, D.: ARTDEFO: Accurate real time deformable objects. In: Proceedings of SIGGRAPH 1999. Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH (1999) 65–72

6. Cotin, S., Delingette, H., Ayache, N.: Real-time elastic deformations of soft tissues for surgery simulation. IEEE Transactions on Visualization and Computer Graphics **5** (1999) 62–73
7. Picinbono, G., Delingette, H., Ayache, N.: Non-linear and anisotropic elastic soft tissue models for medical simulation. In: Proc. IEEE Intl. Conf. Robotics and Automation, Seoul, Korea (2001) 1371–6
8. Müller, M., Dorsey, J., McMillan, L., Jagnow, R., Cutler, B.: Stable real time deformations. In: Proc. ACM SIGGRAPH symposium on Computer Animation, San Antonio, TX (2002) 49–54
9. Székely, G., Brechbühler, C., Hutter, R., Rhomberg, A., Ironmonger, N., Schmid, P.: Modelling of soft tissue deformation for laparoscopic surgery simulation. Medical Image Analysis **4** (2000) 57–66
10. Zhuang, Y., Canny, J.: Real-time simulation of physically realistic global deformation. In: SIGGRAPH99 Sketches and Applications. SIGGRAPH, Los Angeles, California (1999)
11. Wu, X., Downes, M., Goktekin, T., Tendick, F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In Chalmers, A., Rhyne, T.M., eds.: Eurographics 2001, Manchester, UK (2001) Appearing in *Computer Graphics Forum*, vol. 20, no. 3, Sept. 200 1, pp. 349-58.
12. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proceedings of SIGGRAPH 1998. Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH (1998) 43–54
13. Capell, S., Green, S., Curless, B., Duchamp, T., Popovic, Z.: A multiresolution framework for dynamic deformations. In: Proc. ACM SIGGRAPH symposium on Computer Animation, San Antonio, TX (2002) 41–7
14. Zienkiewicz, O., Taylor, R.: The Finite Element Method. fourth edn. McGraw-Hill, London (1994)
15. Debunne, G., Desbrun, M., Cani, M.P., Barr, A.: Dynamic real-time deformations using space and time adaptive sampling. In: Proceedings of SIGGRAPH 2001. Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH (2001) 31–6
16. Grinspun, E., Krysl, P., Schröder, P.: CHARMS: a simple framework for adaptive simulation. In: Proceedings of SIGGRAPH 2002. Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH (2002) 281–90
17. Wesseling, P.: An Introduction to Multigrid Methods. John Wiley, New York (1992)
18. Demmel, J.: Applied Numerical Linear Algebra. SIAM, Philadelphia, PA (1997)
19. Wu, X.: Design of an Interactive Nonlinear Finite Element Based Deformable Object Simulator. PhD thesis, Department of Mechanical Engineering, University of California, Berkeley (2002)
20. Shewchuk, J.: Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In Lin, M., Manocha, D., eds.: Applied Computational Geometry: Towards Geometric Engineering. Springer-Verlag (1996) 203–222
21. Geuzaine, C., Remacle, J.F.: Gmsh: a three-dimensional finite element mesh generator with pre- and post-processing facilities (2002) http://www.geuz.org/gmsh/.
22. Bathe, K.: Finite Element Procedures. Prentice Hall, Englewood Cliffs, NJ (1996)

Appendix

The RaISoV algorithm and the simulation video can be found at http://www.medicalsim.org/People/Xunlei/symposium2004.html.