A semi-implicit method for real-time deformation, topological changes, and contact of soft tissues

Paper id : 269

Abstract. We address in this paper the problem of handling topological changes and contact in surgical simulation in real time. We propose an implicit scheme able to simulate cutting and contact of heterogeneous solids undergoing large displacements and rotations interactively. The method relies on a hybrid CPU/GPU implementation with an asynchronous computation of the preconditioner. We tackle various academic examples as well as the simulated resection of a brain tumour. The salient points of the method are: (1) its insensitivity to the size of stiffness ratios in the model; (2) the ability to use relatively large time steps even in the presence of heterogeneous structures; (3) stability during topological modifications driven by user interaction.

1 Introduction

A series of worldwide directives, such as the "European Working Time Directives", imply that surgeon trainees participate in $30-40\%$ fewer operations than before. This decrease in training time may lead to inadequate preparation for unsupervised practice as consultants. Beyond training, expert surgeons themselves would like to be able to rehearse a complex intervention on simulators. In this paper, surgical simulators are meant as computer-based tools used to simulate a surgeon's interaction with the body.

Developing surgical simulators poses major challenges, including: (i) The need to be patient specific both in the geometric description of the organs and in their physical behaviour. Patient specificity compounds the already significant difficulties associated with robust and automatic medical image segmentation and mesh generation especially when substructures such as capillaries, nerve tracts, fibres, are of interest. (ii) Handling realistically, in real-time and with haptic feedback, the surgeon's interactions with the virtual body: pushing, prodding, palpation, needle insertion, cutting, burning, etc. (iii) A realistic model of the boundary conditions, which usually involve contact, between multiple heterogenous deformable bodies, of variable stiffness, each undergoing large displacements, rotations, or even strains. (iv) Controlling the error, validating the results and evaluating the practical impact of simulators.

In this paper, we focus on (ii) and (iii), whilst the method we propose offers the prospect of error control during simulations (iv). Specifically, we are interested in simulating surgical procedures involving significant topological changes and contact, such as those occurring when performing the resection of a tumour.

2 Background

2.1 Requirements An accepted requirement for realistic soft tissue simulation is to model both geometric and material non-linearities. Material non-linearities are expressed through the non-linearity of the constitutive law relating the strain tensor to the stress tensor, characteristic of soft tissues. Geometric non-linearities are associated with large geometrical transformations, for example, large displacements, large rotations but small strains. Compared to spring-mass and spring-tensor methods, weighted-residual methods are best suited to handle material non-linearities as the constitutive law can be used explicitly. We use here the corotational FEM, introduced by [8] in the graphics community, which is restricted to large displacements and rotations, but only small strains.

Beyond the need to handle non-linearities, our additional working hypothesis is that realistic surgical simulation also requires to interactively compute significant topological modifications of heterogeneous organs, e.g. cutting, and to model faithfully the boundary conditions to which the organs are subjected. These boundary conditions involve contact with both neighbouring anatomical structures and with the surgical instruments. Performing this in real-time is complicated by the fact that the organs have complex geometries and are both geometrically and materially non-linear.

2.2 Formulation Once the spatial discretization is settled, a time integration scheme must be chosen. Most prototype surgical simulators use explicit time integrators, since they require no matrix inversion and are thus computationally cheap and naturally parallelizable [6]. The inherent difficulty in explicit methods however is the choice of the mesh size and time step Δt to guarantee stability. This difficulty is exacerbated in the presence of large stiffness ratios, contact and topological changes. Additionally, the residual of the governing equations cannot be directly controlled and the usual assumption is that if the method is stable, then the error on the satisfaction of equilibrium is "sufficiently low".

We choose a backward Euler implicit time integrator, which is unconditionally stable and offers the possibility of equilibrium error control. The nodal velocities $\dot{\boldsymbol{u}}$ and nodal positions $\boldsymbol{u}$ are updated based on the nodal accelerations $\ddot{\boldsymbol{u}}$ at the end of the time step: $\dot{\boldsymbol{u}}_{t+\Delta t} = \dot{\boldsymbol{u}}_t + \Delta t\, \ddot{\boldsymbol{u}}_{t+\Delta t} \quad \boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \Delta t\, \dot{\boldsymbol{u}}_{t+\Delta t}$. Letting $\mathbf{M}$ be the *mass matrix* obtained from the spatial discretisation, and $\mathbf{f}$ the force function, $\ddot{\boldsymbol{u}}$ can be computed by solving $\mathbf{M} \cdot \ddot{\boldsymbol{u}}_{t+\Delta t} = \mathbf{f}(\boldsymbol{u}_{t+\Delta t}, \dot{\boldsymbol{u}}_{t+\Delta t})$.

As in any implicit scheme, computing $\mathbf{f}$ requires the positions and velocities at the end of the time step, that are unknown. We use a first-order approximation as described in [1], which provides the final linearized system:

$$\underbrace{\left(\mathbf{M} - \Delta t \mathbf{B} - \Delta t^2 \mathbf{K}\right)}_{\mathbf{A}} \underbrace{\Delta \mathbf{v}}_{\boldsymbol{x}} = \underbrace{\Delta t\, \mathbf{f}(\boldsymbol{u}_t, \dot{\boldsymbol{u}}_t) + \Delta t^2\, \mathbf{K}\, \dot{\boldsymbol{u}}_t}_{\boldsymbol{b}}, \tag{1}$$

where $\Delta \mathbf{v} = \Delta t\, \ddot{\boldsymbol{u}}_{t+\Delta t} = \dot{\boldsymbol{u}}_{t+\Delta t} - \dot{\boldsymbol{u}}_t$, and $\mathbf{K}, \mathbf{B}$ are the *stiffness* and *damping* matrices. This set of equations must be solved at each time-step, since $\mathbf{K}, \mathbf{B}$ continuously evolve due to geometrical and material non-linearities. The Conjugate Gradient Method (CGM) is a popular iterative method for this. However,

the condition number of the system matrix $\mathbf{A}$ determines the convergence rate of the CGM and is strongly affected by large ratios in element size or in stiffness through the domain. Both cases potentially occur when simulating heterogeneous materials and when subdividing elements to handle topological changes. A common solution is to precondition $\mathbf{A}$ with a preconditioner, i.e. an approximation of $\mathbf{A}$ which is easier to inverse. Such Preconditioned CGMs improve the conditioning of the system, decreasing the number of iterations to convergence.

Our method relies on a preconditioning technique similar to [2] which updates at a lower frequency, an exact factorization $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$. Such a factorization provides a reasonable approximation of the system matrix at all times during the simulation, so that the CGM only requires a few (2-5) iterations to converge. The method of [2] is further improved by rotating the preconditioning operator with a rotation matrix $\mathbf{R}_{t\to t_u}$, defined as the average nodal rotation matrix computed between the current state and t_u the time of the last factorization $\left(\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}\right)_{t_u}$.

$$\mathbf{P} \;=\; \mathbf{R}^{\mathrm{T}}_{t\to t_u} \quad \left(\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}\right)_{t_u} \quad \mathbf{R}_{t\to t_u}. \tag{2}$$

2.3 Simulation of the interactions Following [5], we use Lagrange multipliers to enforce no-penetration at the end of each time step. Defining $\boldsymbol{\delta}^0$ the initial interpenetration of the time step, $\mathbf{C} = (\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{\Delta t}\mathbf{B} + \mathbf{K})^{-1}$ the *compliance matrix*, $\mathbf{H}$ the contact Jacobian and $\mathbf{W} = \mathbf{H}\mathbf{C}\mathbf{H}^{\mathrm{T}}$ the *Delasus operator*. $\boldsymbol{\delta}$ and $\boldsymbol{\lambda}$ being respectively the contact-free interpenetration vector and the associated contact forces are related by a Linear Complementary Problem (LCP):

$$\text{find } (\boldsymbol{\delta}, \boldsymbol{\lambda}) \text{ such that } \quad \boldsymbol{\delta} = \mathbf{W}\boldsymbol{\lambda} + \boldsymbol{\delta}^0 \text{ subject to the Kuhn-Tucker conditions} \tag{3}$$

For accuracy and stability it is crucial to employ a realistic compliance matrix accounting for the global rotation and displacement of the bodies, but the computation of $\mathbf{C}$ in real-time remains an open problem. It was proposed in [4], to precompute $\mathbf{C}$ from the initial position, and to update it with the Sherman Morrison Formula (SMF) in case of topological modifications, which creates instabilities after a "large" number of modifications, and leads to inaccuracies for large displacements/rotations. Recently, [3] proposed to use the asynchronous preconditioner described in equation (2) to build an approximation of $\mathbf{W}$:

$$\mathbf{W} \;\approx\; \frac{1}{\Delta t^2}\,\mathbf{H}\left(\mathbf{R}\,\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\right)^{-1}\mathbf{H}^{\mathrm{T}}, \tag{4}$$

where the updates of the preconditioner prevent the divergence of the system, and more importantly, the method enables to use large time steps for the contact response. However, no solution was provided to handle topological modifications. Our contribution overcomes this difficulty.

3 Simulation of cutting

3.1 Topological modifications, cutting Cutting or other topological modifications occur locally in the domain. So, rather than reconstructing the overall

mesh, we incrementally update it within 3 steps: first we remove the intersected elements from the current mesh; second we subdivide the removed elements as described in [7]; third, we add the subdivided elements. However, the method is general and can be used to simulate interactively any other topological modification occurring in surgery simulation such as burning and ultrasonic aspiration.

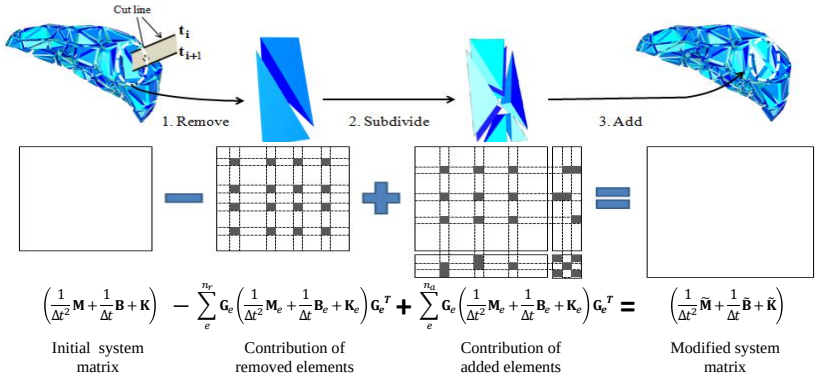

Fig. 1. Incremental update of the mesh structure for the cut.

The subdivision process affects the *stiffness, mass* and *damping* matrices (see fig. 1), and the final linear system $\mathbf{A}$ (i.e. equations (1),(2) and (4)). The CGM used to solve equation (1) only requires to perform matrix vector products. Thus, $\mathbf{A}$ can be directly evaluated from the FE mesh to instantly take into account the modifications in the solution. However, the preconditioner used in equations (2) and (4) is updated with delay, and the modifications significantly affect its efficacy. Moreover, contrary to equation (2) where the PCGM ensures the convergence of the system, the preconditioner is directly used to build an approximation of the contact problem in equation (4). Therefore, the delay of the updates can lead to instabilities and inaccuracies, in particular when treating the contact with the instruments or the self-collisions between different cut parts.

3.2 Low rank update of the Asynchronus preconditioner We propose to use the Sherman Morrison Formula (SMF) to compute the correction of the preconditioner due to the topological changes:

$$\begin{aligned}\widetilde{\mathbf{P}}^{-1} &= (\mathbf{P}+\mathbf{G}\mathbf{N}\mathbf{G}^{\mathrm{T}})^{-1} \\ &= \underbrace{\mathbf{P}^{-1}}_{\text{Last factorization}} - \underbrace{\mathbf{G}\mathbf{P}^{-1}\left(\mathbf{N}^{-1}+\mathbf{G}\mathbf{P}^{-1}\mathbf{G}^{\mathrm{T}}\right)^{-1}\mathbf{P}^{-1}\mathbf{G}^{\mathrm{T}}}_{\text{Correction due to the cut}} \end{aligned} \quad (5)$$

where $\widetilde{\mathbf{P}}$ is the modified preconditioner, $\mathbf{G}$ is a globalization matrix which maps the rows/columns to the global system and $\mathbf{N} = \mathbf{G}^{\mathrm{T}}(\widetilde{\mathbf{P}} - \mathbf{P})\mathbf{G}$ is the perturbation. $\mathbf{N}$ is fast to compute since it only involves subtractions of small matrices associated only with the nodes affected by the cut. Indeed, each new factorization implicitly contains all previous modifications, and an important advantage

of updating the preconditioner is that this suppresses all previous perturbations. The SMF correction is therefore necessary only between the updates of the preconditioner, and the perturbation only involves only the few nodes affected since the last two updates (see fig. 2).

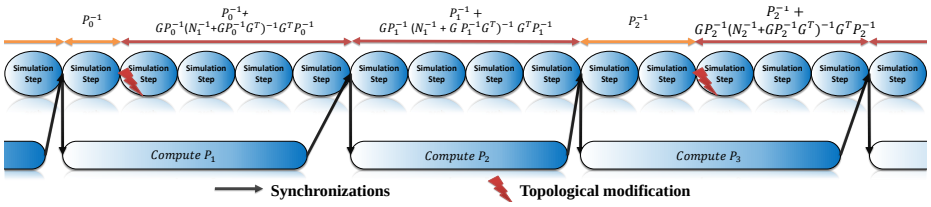

Fig. 2. Correction of the preconditioner during topological modifications. When a modification is performed on the mesh, we first compute the correction of the current factorization. Then we compute the correction of the preconditioner which was being calculated at the time of the cut. After two consecutive updates without topological modification, the preconditioner does not need any additional correction.

Contrary to [4], we do not store the dense inverse of $\mathbf{C}$. Instead we use the sparse factorization of $\mathbf{A}$. The SMF cannot be directly applied with such a factorization because it explicitly requires the inverse of $\mathbf{C}$. Thus, we proceed in two steps: first we compute the correction for the preconditioner, which is only necessary when a new topological modification is performed; then we apply the correction until the next update of the preconditioner.

3.3 Computation of the correction For each new topological modification, the correction is obtained by computing the two following matrices:

$$\mathbf{U} = \left(\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}\right)^{-1}\mathbf{G}^{\mathrm{T}} \tag{6}$$

$$\mathbf{Q} = \left(\mathbf{N}^{-1} + \mathbf{G}\mathbf{U}\right)^{-1} \tag{7}$$

The computation of $\mathbf{U}$ involves the product of the inverse of the preconditioner with the globalization matrix $\mathbf{G}$. It can be achieved by solving column-independently two sparse triangular systems with the columns of $\mathbf{G}$ (as described in [3] for the contact problem). This operation is the most expensive task, and we use a GPU algorithm similar to [3]. It can also be achieved using CUSPARSE.

Equation (7) requires to compute the inverse of two small matrices ($\mathbf{N}$ and $\mathbf{N}^{-1} + \mathbf{G}\mathbf{U}$) which are performed on CPU. $\mathbf{N}$ is the difference between two mechanical matrices and is ill-conditioned. Its inverse is therefore ill-defined, but contrary to [4], our method does not lead to the accumulation of subsequent round off errors. Indeed, $\mathbf{N}$ is erased as soon as a new factorization is released so that these numerical errors do not accumulate over time.

3.4 Application of the correction Once $\mathbf{U}$ and $\mathbf{Q}$ are computed for a given modification, we use them to correct the preconditioner until the next update. However, the local rotations used in equation (2) and (4) vary at each time step,

which prevents including them in the formulation of the correction. In order to use the rotations to improve the efficacy of the preconditioner, we apply them around the corrected formulation. Substituting (5), (6), (7) in (2) gives:

$$\widetilde{\mathbf{P}}^{-1} \approx \mathbf{R}\left(\mathbf{P}^{-1} - \mathbf{U}^{\mathrm{T}}\mathbf{Q}\mathbf{U}\right)\mathbf{R}^{\mathrm{T}}. \tag{8}$$

The rotations are therefore applied around the final corrected solution. To simplify notations we do not carry those rotations in the upcoming equations. For each iteration of the PCGM, the application of the preconditioner is corrected:

$$\begin{aligned} \widetilde{\mathbf{P}}\boldsymbol{x} = \boldsymbol{b} \Leftrightarrow \boldsymbol{x} &= \left(\mathbf{P}^{-1} - \mathbf{U}^{\mathrm{T}}\,\mathbf{Q}\,\mathbf{U}\right)\boldsymbol{b} \\ \boldsymbol{x} &= \underbrace{\mathbf{P}^{-1}\boldsymbol{b}}_{\boldsymbol{x}^{\mathrm{a}}} - \underbrace{\mathbf{U}^{\mathrm{T}}\,\mathbf{Q}\,\mathbf{U}\,\boldsymbol{b}}_{\boldsymbol{x}^{\mathrm{c}}}, \end{aligned} \tag{9}$$

where $\boldsymbol{x}^{\mathrm{c}}$ is the correction of the solution, which involves 3 dense matrix–vector products, and are performed on GPU using the CUBLAS library. $\boldsymbol{x}^{\mathrm{a}}$ is equivalent to applying the preconditioner without correction as in equation (2), except that if nodes are added during the subdivision process, $\mathbf{P}$ is padded with the identity:

$$\begin{pmatrix} \boldsymbol{x}^{\mathrm{a}}_m \\ \boldsymbol{x}^{\mathrm{a}}_a \end{pmatrix} = \begin{pmatrix} \mathbf{P}^{-1} & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_m \\ \boldsymbol{b}_a \end{pmatrix}, \tag{10}$$

where subscripts m and a correspond to the modified and added nodes. This approach is equivalent to assuming that the added degrees of freedom were present before the cut, but not attached to the mechanical system.

For the contact response, the corrected *Delasus operator* $\mathbf{W}$ is obtained with:

$$\begin{aligned} \mathbf{H}\widetilde{\mathbf{P}}\mathbf{H}^{\mathrm{T}} &= \mathbf{H}\left(\mathbf{P}^{-1} - \mathbf{U}^{\mathrm{T}}\,\mathbf{Q}\,\mathbf{U}\right)\mathbf{H}^{\mathrm{T}} \\ &= \underbrace{\mathbf{H}\mathbf{P}^{-1}\mathbf{H}^{\mathrm{T}}}_{\mathbf{W}^{\mathrm{a}}} - \underbrace{\mathbf{H}\,\mathbf{U}^{\mathrm{T}}\,\mathbf{Q}\,\mathbf{U}\,\mathbf{H}^{\mathrm{T}}}_{\mathbf{W}^{\mathrm{c}}}, \end{aligned} \tag{11}$$

where $\mathbf{W}^{\mathrm{a}}$ is the asynchronous *Delasus operator*, and $\mathbf{W}^{\mathrm{c}}$ its correction. As in equation (10), $\mathbf{H}$ may involve constraints on the newly created degrees of freedom, so that we pad $\mathbf{P}$ with the identity matrix, and $\mathbf{W}^{\mathrm{a}}$ is computed by:

$$\mathbf{W}^{\mathrm{a}} = \left(\mathbf{H}_m\ \mathbf{H}_a\right)\begin{pmatrix} \mathbf{P}^{-1} & 0 \\ 0 & \mathbf{I} \end{pmatrix}\begin{pmatrix} \mathbf{H}_m \\ \mathbf{H}_a \end{pmatrix} = \begin{pmatrix} \mathbf{H}_m\mathbf{P}^{-1}{\mathbf{H}^{\mathrm{T}}}_m & 0 \\ 0 & \mathbf{H}_a{\mathbf{H}_a}^{\mathrm{T}} \end{pmatrix} \tag{12}$$

$\mathbf{H}_m\mathbf{P}^{-1}{\mathbf{H}^{\mathrm{T}}}_m$ is solved using the GPU-based algorithm introduced in [3], whereas $\mathbf{H}_a{\mathbf{H}_a}^{\mathrm{T}}$ involves a sparse matrix product which is parallelized using the CUSPARSE library. Finally, $\mathbf{W}^{\mathrm{c}}$ is obtained by 3 small dense matrix products which are performed on GPU using the CUBLAS library.

4 Results

We compare our method (**SMF**) with a standard **CGM** on a simulation composed of a beam cut lengthwise and falling under gravity (see table 3). **Build**

is the time to assemble **A**: For the CGM, the matrix is directly evaluated from the mesh structure at each iteration, whereas the preconditioner is updated on average every 4.30 simulation steps, each update requiring to fully assemble **A** to perform the factorization in equation (2). **Iterations** and **Solve** are respectively the number of iterations per time step, and the corresponding time to solve the system with a tolerance at 10^{-7}. The method significantly decreases the number of iterations, and provides an average speed-up of 2.6× compared to the **CGM**.

Nodes	Method	Affected Nodes	Build (ms)	Itera-tions	Solve (ms)	Total (ms)
1200	CGM	NA	0.01	489.40	49.50	50.99
	SMF	46.39	7.30	6.59	10.41	20.25
1600	CGM	NA	0.01	584	60.71	62.47
	SMF	52.50	9.00	7.26	15.70	27.57
2000	CGM	NA	0.01	687.29	76.29	78.38
	SMF	54.75	10.24	7.43	21	34.40

Fig. 3. Performances and convergence comparison.

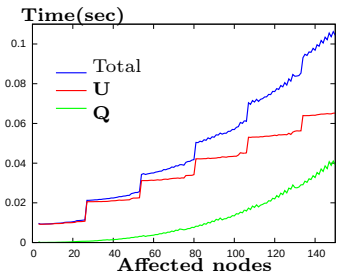

Fig. 4. Overhead of the SMF.

Computing **U** and **Q** is the main overhead of our method (see fig. 4) and represents an average of 15% of the computation time of a typical time step on the beam example in fig 3. However, these operations are performed only when a new topological modification is detected (i.e. every 3.4 simulation steps on the beam example). The computation of **U** is the most expensive task, and its GPU parallelization[1] is the key point to enable real-time computations. The inversion of **Q** is inexpensive for small perturbations but quickly becomes costly for large perturbations. In practice, the number of affected nodes remains very small since the preconditioner is updated several times per second. The application of the correction (i.e. applying $\boldsymbol{x}^c$ in equation (9) and $\mathbf{W}^c$ in (11)) is negligible since it represents less than 1% of the computation time of a time step.

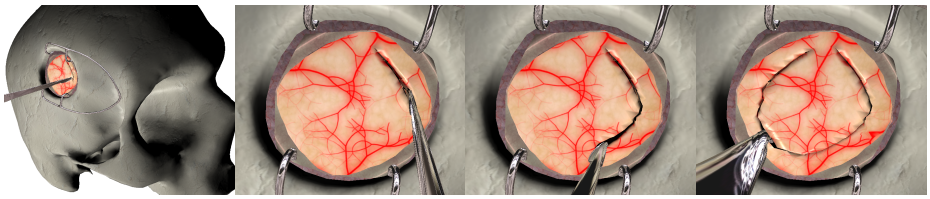

Fig. 5. Real-time simulation of a brain tumor resection.

[1] The staircase aspect of the curve is a consequence of the GPU parallelization where up to 25 nodes can be processed simultaneously to compute the columns of **U**

We applied our method to the simulation of the resection of a brain tumour. The brain is modeled as a heterogeneous deformable body, composed of $1,734$ nodes and $7,680$ linear tetrahedral elements. The tumor is $20\times$ stiffer than the brain. During the simulation, the preconditioner is updated every 5.6 steps, and a new topological modification appears every 5.5 simulation steps, affecting 24 nodes. A total of 553 modifications are performed, and the method remains stable with an average of 5.70 iterations to solve the linear system. The collisions and self-collisions are correctly solved while processing the modifications, and cut parts can instantaneously be separated upon contact with the instrument. Finally, we achieve between 20 and 40 FPS and the method remains interactive.

5 Conclusions and future work

We proposed an implicit method to simulate topological modifications at interactive rates in heterogeneous solids whilst resolving existing or new contact surfaces emanating from these modifications. The key point of this hybrid CPU/GPU algorithm is an accurate update of the compliance matrix relying on an asynchronous preconditioner and SM corrections. The method is stable and efficient regardless of stiffness ratios as indicated by a simulated tumour resection. This contribution is a way towards " error control" at each step. To treat larger problems, mesh adaptivity, enrichment and model reduction are fruitful avenues. Finally, material non-linearities should be considered, which is on-going work.

References

1. D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1998.
2. H. Courtecuisse, J. Allard, C. Duriez, and S. Cotin. Asynchronous preconditioners for efficient solving of non-linear deformations. In *Proceedings of Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, nov 2010.
3. H. Courtecuisse, J. Allard, C. Duriez, and S. Cotin. Preconditioner-based contact response and application to cataract surgery. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, sept 2011.
4. H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D. Y. Lee, and S. Cotin. GPU-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in Biophysics and Molecular Biology*, 2011. Special Issue on Soft Tissue Modelling.
5. C. Duriez, F. Dubois, A. Kheddar, and C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):36–47, 2006.
6. G. Joldes, A. Wittek, and K. Miller. Real-time nonlinear finite element computations on gpu–application to neurosurgical simulation. *Computer methods in applied mechanics and engineering*, 199(49):3305–3314, 2010.
7. A. B. Mor and T. Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Proceedings of MICCAI*, pages 598–607, 2000.
8. M. Müller and M. Gross. Interactive virtual materials. In *GI '04: Proc. of Graph Interface*, pages 239–246. Canadian Human-Computer Communications Society, 2004.