Research Report of Intelligent Systems for Medicine Laboratory

Report # ISML/04/2006, May 2006

Mathematical Methods for Computational Soft Tissue Medical Imaging

Grand Roman Joldes

Intelligent Systems for Medicine Laboratory
School of Mechanical Engineering
The University of Western Australia
35 Stirling Highway
Crawley WA 6009, AUSTRALIA
Phone: + (61) 8 6488 1901
Fax: + (61) 8 6488 1024
Email: grandj@mech.uwa.edu.au
http://www.mech.uwa.edu.au/ISML/

Abstract

This report covers the work performed since the submission of the research proposal in December 2005.

Biomechanical models have been used for quite some time for image registration. These models try to balance the accuracy of the computed results with the simulation runtime. A fast simulation requires major simplifications in the model and this has a negative impact on the simulation accuracy.

The report presents different algorithms for the computation of the steady state solution in a finite element simulation using static or cvasistatic simulations. These algorithms are designed to increase the simulation speed without reducing the accuracy of the results. Therefore they can handle both geometric and material non-liniarities.

One way of increasing the computation speed is the using of simple linear elements when constructing the mesh, such as the linear tetrahedron and the linear under-integrated hexahedron. Both these elements can have a negative impact on the accuracy of the results because of volumetric locking and hourglassing. Fast algorithms that improve the behavior of these elements will be presented.

Keywords: biomechanics, soft tissue, Total Lagrangian, explicit integration, non-locking tetrahedral, hourglass control

Table of content

1. Introduction

There is wide international concern about the cost of meeting rising expectations for health care, particularly if large numbers of people require currently expensive procedures such as brain surgery. Costs can be reduced by using improved machinery to help surgeons perform these procedures quickly and accurately, with minimal side effects. A novel partnership between surgeons and machines, made possible by advances in computing and engineering technology, could overcome many of the limitations of traditional surgery. By extending the surgeons' ability to plan and carry out surgical interventions more accurately and with less trauma, Computer-Integrated Surgery (CIS) systems could help to improve clinical outcomes and the efficiency of health care delivery. CIS systems could have a similar impact on surgery to that long since realized in Computer-Integrated Manufacturing (CIM).

Mathematical modeling and computer simulation have proved tremendously successful in engineering. Computational mechanics has enabled technological developments in virtually every area of our lives. One of the greatest challenges for mechanists is to extend the success of computational mechanics to fields outside traditional engineering, in particular to biology, biomedical sciences, and medicine [1]. The goal of surgical simulation research is to model and simulate deformable materials for applications requiring real-time interaction. Medical applications for this include simulation-based training, skills assessment and operation planning. A surgical simulator must predict the deformation field within the organ, so that it can be displayed to the user, and the internal forces (stresses), so that reaction forces acting on surgical tools can be computed and conveyed to the user through haptic feedback.

Biomechanical models are developed for predicting the organ deformation using the finite element method, but many such models are simplified in order to decrease the computational effort, e.g. they consider only infinitesimal deformations and/or linear material laws. These simplifications have great influence on the accuracy of the obtained results in a finite element analysis, inducing significant errors [2-4]. Therefore material law and geometric non-liniarities must be considered when a solution method is chosen.

There are three methods of reducing the computation time: improvement of the algorithms, usage of faster hardware or usage of parallel computing. We will concentrate on the first method, as the usage of faster hardware is limited by the existing technology and the usage of parallel computing leads to more complex and more expensive hardware and software systems.

2. Finite Element Algorithms

When designing a finite element solution method there are many aspects that must be considered, such as the formulation used (Total or Updated Lagrangian), solution scheme and the type of elements used for constructing the mesh. We will discuss these aspects in the following sections.

2.1. The Total Lagrangian Formulation

Various spatial discretisation schemes are possible while using the finite element method [5]. The algorithms used by the great majority of commercial finite element programs use the Updated Lagrangian formulation, where all variables are referred to the current (i.e. from the end of the previous time step) configuration of the system (Ansys [6], ABAQUS [7],

ADINA [8], LS Dyna [9], etc.). The advantage of this approach is the simplicity of incremental strain description. The disadvantage is that all derivatives with respect to spatial coordinates must be recomputed in each time step, because the reference configuration is changing. The reason for this choice is historical – at the time of solver development the memory was expensive and caused more problems than the actual speed of computations. The first key idea in the finite element algorithms development was to use the Total Lagrangian formulation of finite element method, where all variables are referred to the original configuration of the system. Second-Piola Kirchoff stress and Green strain are used. The decisive advantage of this formulation is that all derivatives with respect to spatial coordinates are calculated with respect to original configuration and therefore can be precomputed. The proposed stress and strain measures are appropriate for handling geometric non-liniarities (finite deformations).

The usage of Total Lagrangian explicit integration for simulating physically realistic deformations was also proposed in [10]. A method for decreasing the computation time when using non-linear elasticity was presented in [11], but it only works for tetrahedral meshes and special elastic material laws.

Because biological tissues behavior can be described in general using hyper-elastic or hyper-visco-elastic models [12], the usage of the Total Lagrangian formulation also leads to a simplification of material law implementation as these material models can be easily described using the deformation gradient.

2.2. The mesh

Before any simulation of an organ deformation can be carried out using the finite element method, a mesh must be constructed based on the geometry of the organ. Because of the computation time requirement, the mesh should be constructed using low order elements that are not computationally intensive, such as the linear tetrahedron or the linear under-integrated hexahedron. The standard formulation of the tetrahedral element exhibits volumetric locking [13], especially in case of soft tissues such as the brain, that are modeled as almost incompressible materials [14]. Therefore hexahedral elements are preferred.

Many algorithms are now available for fast and accurate automatic mesh generation using tetrahedral elements, but not for automatic hexahedral mesh generation [15-17]. This is one reason why many authors proposed the usage of tetrahedral meshes for their models [18-21]. In order to automate the simulation process, mixed meshes (having both hexahedral and tetrahedral elements) with predominantly hexahedral elements are the most convenient.

The under-integrated hexahedral elements require the usage of an hourglass control algorithm in order to eliminate the zero energy modes which arise from the one-point integration [22]. Special algorithms for handling hourglass control for the hexahedral elements and to reduce locking of the tetrahedral elements must therefore be implemented.

2.3. Hourglass control

One of the earliest and most popular hourglass control algorithms, that is currently available in many commercial software finite element packages (ABAQUS [23], LS-DYNA [9]) is the one proposed by Flanagan and Belytschko in [22], also known as the perturbation

hourglass control. This method is applicable for hexahedral elements with arbitrary geometry used for the simulation of large deformations (including rigid body motions).

Several other algorithms were proposed for hourglass control. Liu et al. proposed the usage of stabilization matrices for hourglass control in the so called physical stabilization method [24]. The algorithm required the storage of 36 hourglass stresses in addition to the single-point stresses and the resulting stabilization forces causes the element to experience volumetric and shear locking [25].

Belytschko and Bindeman proposed an assumed strain stabilization method [26] and Puso combined the physical stabilization method with the assumed strain method in order to obtain an efficient enhanced assumed strain physically stabilized element [25]. These methods provide an improved behavior of the element in bending dominated problems. However, the stability of these methods can not be guaranteed for general deformation states and arbitrarily shaped elements [27]. The enhanced strain physically stabilized element is also available in the commercial finite element codes [9, 23].

In the context of large deformations, rigid body motions and arbitrarily shaped elements, the perturbation hourglass control is the most computationally efficient of the presented hourglass control methods. The increased performance of the enhanced assumed strain hourglass control methods in bending dominated problems is of little importance in surgical simulation.

Starting from that algorithm proposed by Flanagan and Belytschko we could show that the Total Lagrangian formulation is also recommended from the point of view of efficient hourglass control implementation, as many quantities involved can be pre-computed. We have shown in [28] that the hourglass control forces for each element can be computed (in matrix form) as:

$$ {}^{t}_{0}\mathbf{F}^{Hg} = k\,{}_{0}\boldsymbol{\gamma}\,{}_{0}\boldsymbol{\gamma}^{T}\,{}^{t}_{0}\mathbf{u} \qquad (\mathbf{1}) $$

where k is a constant that depends on the element geometry and material properties, $\boldsymbol{\gamma}$ is the matrix of hourglass shape vectors and $\mathbf{u}$ is the matrix of current displacements. The notation from [13] is used, where the left superscript represents the current time and the left subscript represents the time of the reference configuration, which is 0 for Total Lagrangian. In (1) all quantities except $\mathbf{u}$ are constant and can be pre-computed, making the hourglass control mechanism very efficient.

Several simulations were conducted using the proposed hourglass control mechanism. In order to assess the accuracy of the computation results, they were compared with the results obtained using mixed formulation fully integrated elements and the commercial finite element software (ABAQUS). A very good agreement of the results was obtained [28]. Our under-integrated hexahedral element using perturbation based stiffness hourglass control is almost five times more computationally efficient in the Total Lagrangian framework than in the Updated Lagrangian framework.

2.4. Improved tetrahedral elements

There are a number of improved (non-locking) linear tetrahedral elements already proposed by different authors [29-31]. These formulations are either much more computationally intensive than the standard formulation or the volumetric locking control mechanism depends on material properties (e.g. bulk modulus), making harder the interfacing

of different materials. Our volumetric lock control mechanism is computationally inexpensive and depends solely on kinematic variables.

As defined in [30], the nodal Jacobian is the ratio between the current and initial nodal volumes. The nodal volume is computed as a sum of fractions of the surrounding element volumes.

Using the nodal Jacobians, an average Jacobian can be computed for each element. Because the element Jacobian is equal to the determinant of the element deformation gradient, we define a modified element deformation gradient that has the same isochoric part as the normal deformation gradient, but the volumetric part is modified so that its determinant (and therefore the volumetric deformation) is equal to the average element Jacobian.

The computation of the nodal forces (or stiffness matrix) can now be done in the usual manner, but using the modified deformation gradient instead of the normal deformation gradient for defining the strains. This way there is no need for computing the isochoric and deviatoric components of the internal forces separately, and the existing material law can be used [32].

A similar approach is used in ABAQUS [7] for defining the selectively reduced integration of the volumetric term for the fully integrated first-order isoparametric hexahedral elements, in order to prevent locking in incompressible or nearly incompressible cases. The difference is that while for the selectively reduced integrated hexahedra the Jacobians are averaged over an element, in our case the Jacobians are averaged between elements sharing the same node.

The performance of the proposed formulation is evaluated using the Total lagrangian Explicit Dynamics (TLED) algorithm presented in [33] against the standard tetrahedral element formulation. The differences in computed displacements and reaction forces due to volumetric locking are reduced with more than 50%.

The modified TLED code using the improved element is about 40% more computationally expensive per time step than the standard one. This percentage might seem high, but it is only due to the fact that our definition of the standard tetrahedral element using a Total Lagrangian formulation is already very efficient. Compared with Abaqus explicit, the modified TLED implementation is still more than 20 times more efficient. Apart for this, the stability analysis shows that a much higher time step can be used with the improved tetra element and therefore the overall computation time can be reduced.

3. Static analysis algorithms

The static solution of a nonlinear analysis is described by the nonlinear system of equations:

$$\boldsymbol{R}(\boldsymbol{U}) - \boldsymbol{F}(\boldsymbol{U}) = 0 \qquad (\mathbf{2})$$

where $\boldsymbol{R}$ is the vector of externally applied nodal loads and $\boldsymbol{F}$ is the vector of nodal point forces. Both $\boldsymbol{R}$ and $\boldsymbol{F}$ depend nonlinearly on the vector of nodal point displacements $\boldsymbol{U}$.

The most frequently used schemes for the solution of (2) are the Newton-Raphson iterative methods [13, 34, 35], in which we solve for i = 1,2,3…

$$dR^{i-1} = R^{i-1} - F^{i-1} \qquad (\mathbf{3})$$

$$K^{i-1} dU^{i} = dR^{i-1} \qquad (4)$$

$$U^i = U^{i-1} + dU^i \tag{5}$$

Initial conditions:

$$U^0 = U0 \qquad F^0 = F0 \tag{6}$$

where

$$K^{i-1} = \left.\frac{\partial (F - R)}{\partial U}\right|_{U=U^{i-1}} \tag{7}$$

is the current tangent stiffness matrix.

These equations are obtained using a Taylor series expansion of the system of equations about the conditions at iteration i-1. In each iteration we calculate in (3) an out of balance load vector that determines an increment in displacements given by (4). The iterations are continued until an appropriate convergence criterion is satisfied.

The Newton-Raphson method has the property that if the current solution iterate is sufficiently close to the solution and if the tangent stiffness matrix does not change abruptly, we can expect rapid (quadratic) convergence [36, 37].

3.1. Static condensation

The purpose of the analysis is to compute the nodal displacements. These can be partitioned into unknown displacements and known displacements (the displacements of the encastred and displaced nodes). The known displacements are the only ones that produce external nodal loads. These displacements can be eliminated from the system of equations (4).

The static condensation of the known displacements has the following advantages [13]:

a. Reduces the size of the equation system
b. Reduces the storage space required for the assemblage of the tangent stiffness matrix
c. Improves the condition number of the system matrix.

3.2. Alternative solution methods

The major computation cost per iteration for the full Newton-Raphson method lies in finding the solution of the linear system of equations (4), which implies the computation and factorization of the tangent stiffness matrix. Modifications of the method have been proposed that try to reduce this cost [13]:

- The modified Newton-Raphson iteration - the tangent stiffness matrix is computed and factorized only at accepted equilibrium configurations (not at the beginning of each iteration) ;
- Quasi-Newton methods – the inverse of the tangent stiffness matrix is updated between iterations.

Because the material models for soft tissues are only mildly nonlinear (hyper-elastic or hyper-visco-elastic models [12]), the tangent stiffness matrix does not change abruptly, and therefore the modified Newton-Raphson iteration method is a good alternative for reducing the computation time.

The tangent stiffness matrix is a sparse, symmetric banded matrix. This has major implications for the way it is stored and handled, in order to fit in the core memory (RAM) of the computer. The inverse of a sparse banded matrix does not share the same properties, being a full matrix. Even for a few thousands degrees of freedom, such a matrix is too big to fit into the core memory, and requires special algorithms for handling it using backup memory (hard drive). We tried to overcome this problem by using a sparse approximate inverse (SPAI) of the matrix [38], but the computation time for obtaining the SPAI was too high for the method to be effective. Therefore we did not consider the quasi-Newton method as a viable alternative.

3.3. Proposed solution methods

As mentioned before, the major computation cost per iteration for the Newton-Raphson method lies in solving the linear system of equations (4). Nevertheless, the solution of this system of equations provides only an update for the solution of the non-linear system, and therefore it is not necessary to be very exact.

An approximate solution for the linear system of equations (4) can be obtained using an iterative method. Because the tangent stiffness matrix is symmetric and positive definite [13], the convergence conditions for different iterative methods are met and convergence can be guaranteed [39].

By combining the Newton iterations with the iterative methods for solving linear systems of equations the so called composite methods are obtained. An example of such a method is the m-step Newton-SOR process, in which one step of the Newton iteration is combined with m secondary iterations of the SOR (Successive Over Relaxation) method. The theory describing these methods and their convergence properties can be found in [36]. The number of secondary iterations, m_k, can vary between different Newton iterations k, and if $\lim_{k\to\infty} m_k = +\infty$, the rate of convergence of these methods is super-linear.

Although the mathematical aspects of these methods have been studied a long time ago, we have no knowledge of any attempts of using them for solving non-linear systems of equations obtained using the finite-element method. For other proposed applications of these methods, see [40]. In [41] a modified SOR-Newton algorithm is presented, for solving nonlinear algebraic systems resulted from the discretizations of elliptic boundary value problems.

3.4. Improving the convergence using line search

For any Newton iterations, the convergence of the method can be improved by using the line search procedures [42, 43]. By integrating this procedure into a composite method, the algorithm for executing one iteration step becomes:

Compute the tangent stiffness matrix K^{i-1} and the residual forces F^{i-1};

Obtain an approximate solution for the linear equation system

$$K^{i-1}\Delta U^{i-1} = F^{i-1} \qquad (8)$$

by performing m_i iterations using the chosen iterative method;

Update the displacements using:

$$U^i = U^{i-1} + \beta \cdot \Delta U^{i-1} \tag{9}$$

where β is determined so that the projection of the residual force on the search direction is made zero:

$$F^i \Delta U^{i-1} = 0 \text{ , with } F^i = F(U^{i-1} + \beta \cdot \Delta U^{i-1}) ; \tag{10}$$

Equation (10) is obtained by imposing that the norm of the residual is minimized on the search direction:

$$\frac{\partial \left\| F(U^{i-1} + \beta \cdot \Delta U^{i-1}) \right\|_2}{\partial \beta} = 0 \tag{11}$$

The step size β is computed by solving the nonlinear equation (10). In our implementation, this equation is solved using Brent's root finding algorithm that combines the bisection method, the secant method and inverse quadratic interpolation [44]. Finding β to a high accuracy is not efficient, and therefore only an inexact line search is performed, where the number of iterations in the root finding algorithm is limited. Also, the interval in which β can vary is defined. In our implementation this interval is restricted to [0.2, 1].

3.5. Implementation

In order to assess the performances of the proposed composite methods, the following methods were implemented:

- The Newton-Raphson method
- The modified Newton method
- Several composite Newton methods

In our implementation we used the MUMPS direct solver (sequential version) [45] to solve the linear system of equations (12) for the Newton-Raphson and modified Newton methods. This is one of the best direct solvers available, as shown by the results of this comparison report [46].

The solution process was divided in two stages, keeping in mind the application requirements for these algorithms:

- Analysis – this can be done pre-operatively, so it is not time critical. During this stages the following steps are taken:
    - Compute and reserve the necessary memory – the degrees of freedom (DOFs) are reordered so that the memory requirement for the tangent stiffness matrix is reduced. This reordering is done using the reverse Cuthill-McKee algorithm [47-49].
    - Elements generation in memory
    - Association between element local DOFs and the global DOFs
    - Marking of the DOFs that must be condensed out for each element

  - The initial volume and shape functions derivatives are computed for each element
  - The best iterative method is chosen based on the initial tangent stiffness matrix (only for the composite methods)
- Solution – this is the time critical part of the method, so the computation time performances will be measured. This stage consists in the implementation of the different Newton iterative methods, having the following steps:
  - Application of node displacements
  - Computation of the tangent stiffness matrix and residual forces
  - Computation of displacements variation (eq. 15-17)
  - Convergence criteria assessment

In the commercial software the solution step also contains a self-adaptive procedure that chooses the load steps size, because this is the only method to ensure that the starting point is close enough to the solution so that the convergence criteria are met [13]. The convergence criteria are very hard to assess in practice, but usually divergence is very quickly detected in the iterations [37] and the load step size can be modified, for example as proposed in [34]. Our implementation is much simpler, as it is made for comparison purposes only, so the size of the load steps can only be configured by specifying the number of load steps.

For the implementation of the composite Newton methods, the following iterative solvers, available on the public domain, were considered:

- From the Iterative Methods Library (IML++) [50, 51] package, written using templates in C++ on top of SparseLib++[52, 53]:
  - Conjugate Gradient (CG) with diagonal preconditioning
  - Conjugate Gradient (CG) with incomplete Cholesky (IC) preconditioning
- From ITPACK 2C - A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods [54, 55]:
  - Jacobi using Conjugate Gradient (CG) acceleration (JCG)
  - Jacobi using Chebyshev (Semi-Iteration) acceleration (JSI)
  - Successive Over Relaxation (SOR)
  - Symmetric SOR using CG acceleration (SSORCG)
  - Symmetric SOR using Chebyshev acceleration (SSORSI)

3.6. Computational results

Two test problems were considered in order to compare the performances of the composite Newton methods with those of the Newton-Raphson and modified Newton methods: compression of a cylinder and deformation of an ellipsoid. The size of the meshes is in the range of 0.1 – 0.2 meters, close to the sizes of soft organs such as brain.

Both meshes consists of hexahedral elements and were generated using Hypermesh (Fig. 1). We used a Neo-Hookean material, with properties similar to those of the human brain [56] (Young's modulus in un-deformed state equal to 3000 Pa, Poisson's ratio 0.49 and mass

density of 1000 kg/m^3) and fully integrated hexahedrons with selectively reduced integration of the volumetric term, as these elements perform better in case of almost incompressible materials [7].

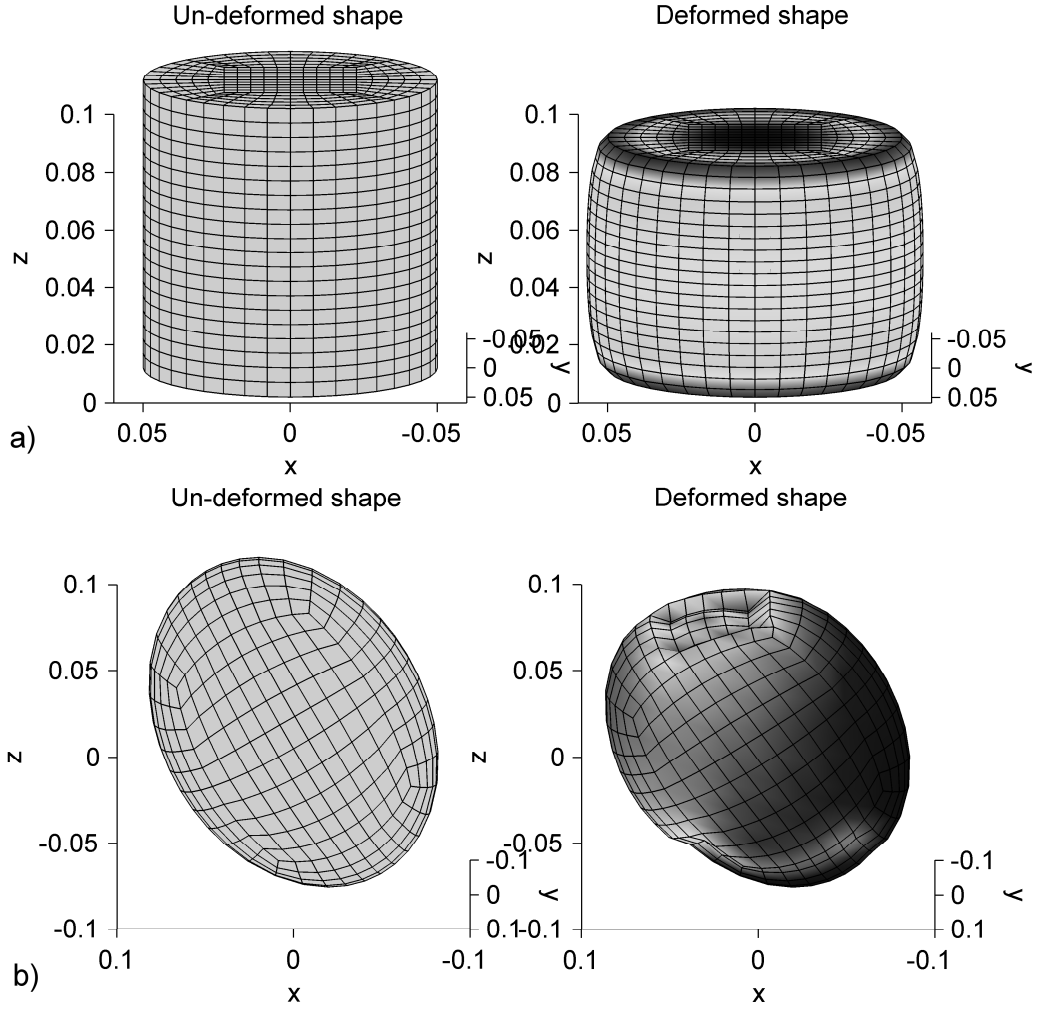


Fig. 1. The un-deformed and deformed meshes

a) Cylinder b) Ellipsoid

The boundary conditions were specified as follows:

- Encastred nodes ($\Delta x = \Delta y = \Delta z = 0$) on the lower surface of the body
- Displaced nodes ($\Delta x = \Delta y = 0$, $\Delta z = d$) on an upper surface of the body, with the maximum displacement d=0.03 m.

In both cases the displacements were applied in only one load step.

3.6.1. Choosing the best iterative method

In the analysis step, when the initial tangent stiffness matrix is computed and the initial load step can be applied, a first linear system of equation of type (4) is obtained. One of the characteristics of the iterative methods is that their performances are problem specific [39, 54], and therefore one can try to choose the best method for the problem that must be solved.

A particularity of the problems that we study is that the structure of the tangent stiffness matrix does not change during the Newton iteration process, and also the values are not expected to change drastically, as the material laws are only mildly non-linear. Therefore, by selecting the best iterative method for the initial system of equations, we can expect also a good behavior for the remaining Newton iterations.

If we rewrite the initial system of equations as:

$$KU = F \tag{12}$$

we can study the convergence of an iterative solver by computing the decrease in the relative residual with the number of iterations. Because we are also interested in the solution time for the solver, we can also study the variation of the relative residual over computation time. The relative residual is given by:

$$r_i = \frac{\|KU_i - F\|_2}{\|F\|_2} \tag{13}$$

where U_i is the approximate solution of system (12) after performing i iterations of the iterative solver.

By plotting the computation time versus number of iterations we can see that for each solver the computation time per iteration is almost constant (there are differences mainly because of the time measurement precision), but some solvers have a higher computation time per iteration than others (Fig. 2). This is true especially for the CG solvers and can be partly explained by the fact that these solvers were implemented in a high level language (C++) and did not use any optimized Sparse Basic Linear Algebra Subprograms (SBLAS) routines.

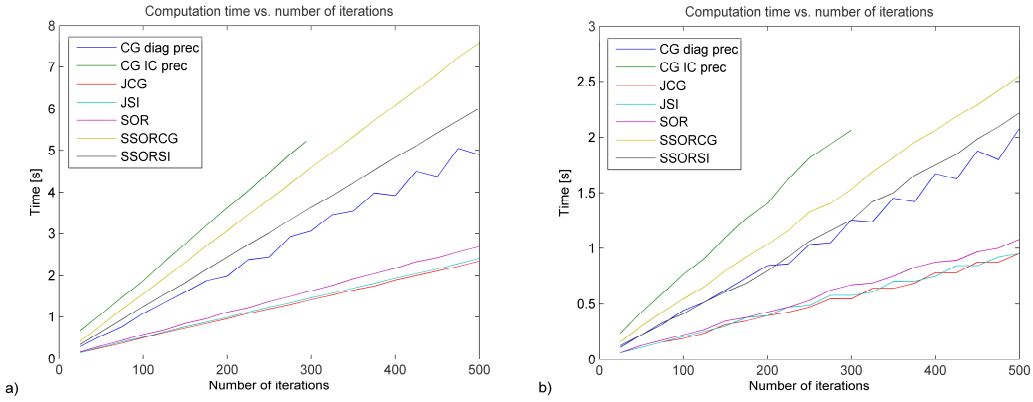


Fig. 2. Computation time for different iterative methods

a) Cylinder b) Ellipsoid

From the plot of relative error versus computation time we can select the best iterative method for the considered problem (Fig. 3). If we consider the best computation time in order to get a relative residual of less that 0.1%, we can see that the JCG solver is the best choice for the

cylinder compression and the SSORSI solver is the best choice for the ellipsoid deformation problem (even if its performances were the worst for the other problem).

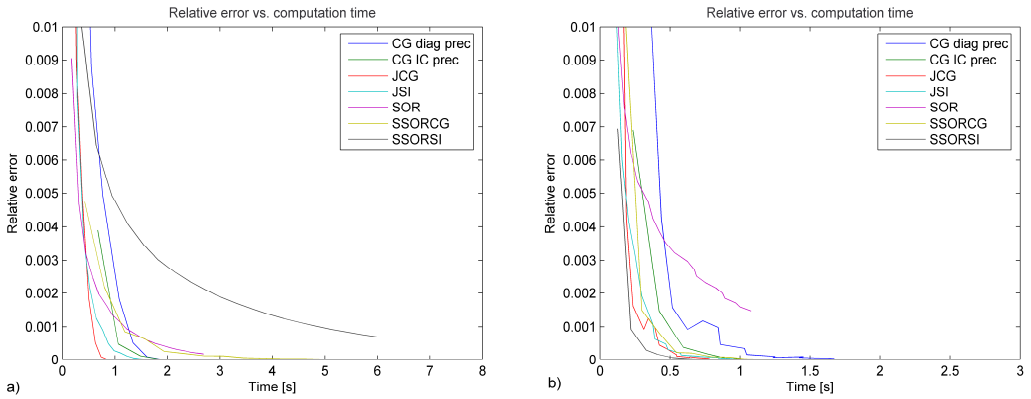


Fig. 3. Relative error versus computation time for different iterative methods

a) Cylinder b) Ellipsoid

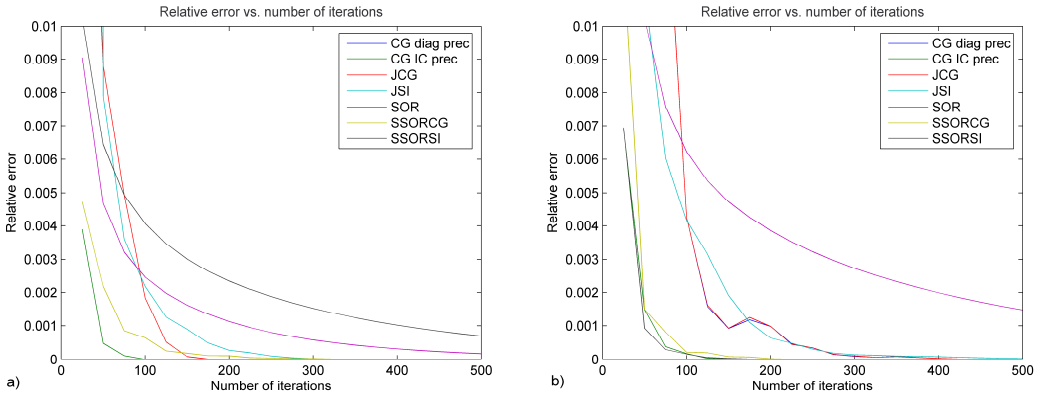


Fig. 4. Relative error versus number of iterations for different iterative methods

a) Cylinder b) Ellipsoid

Another configuration that must be made is the number of iterations that must be performed in order to get an approximate solution to (12). This decision can be made by studying the variation of the relative residual versus the number of iterations for the chosen iterative method (Fig. 4). We can see that all the iterative methods considered have an asymptotic convergence towards the solution. The convergence rate is very good in the first iterations, where the truncation errors dominate the solution, but finally decreases as the computational error becomes the dominant part of the solution, until further iterations do not give any further improvement in accuracy [37]. By studying the curves, we can make a selection of the number of iterations that provides the best convergence rate. For example, we can select 150 iterations for cylinder compression using JCG and 120 iterations for ellipsoid deformation using SSORSI. We can see that this number of iterations not only ensure a good convergence rate, but also a relative residual of less than 0.02%.

3.6.2. Convergence criteria

There are three convergence criteria usually presented in the finite element literature: based on displacements, on residual forces or on internal energy [13, 35].

When displacements are used, the convergence criteria can be written as:

$$\varepsilon_D^i = \frac{\left\| U_a^i - U_a^{i-1} \right\|_2}{\left\| U_a^i \right\|_2} \le \varepsilon_{DLim} \tag{14}$$

This requires that the displacement variation at the end of the iteration be within a certain tolerance of the true displacement solution, where the true displacement solution is approximated with the displacement at the end of the iteration.

The convergence criteria based on residual forces requires that out-of-balance load vector be within a preset tolerance of the original load increment, and can be written as:

$$\varepsilon_F^i = \frac{\left\| F_{ac}^i \right\|_2}{\left\| F_{ac}^0 \right\|_2} \le \varepsilon_{FLim} \tag{15}$$

The third criterion combines the previous two, and compares the increment in internal energy during the iteration (the amount of work done by the out-of-balance loads on the displacement increments) to the initial internal energy increment:

$$\varepsilon_E^i = \frac{\left(U_a^i - U_a^{i-1}\right) F_{ac}^{i-1}}{\left(U_a^1 - U_a^0\right) F_{ac}^0} \le \varepsilon_{ELim} \tag{16}$$

The main problem is to choose the tolerances ε_{DLim}, ε_{FLim} or ε_{ELim} that ensures that the solution has converged within the desired accuracy. If the tolerances are too loose the error in the computed solution will be too high and if the tolerances are very small, the iterations might never converge or the computational cost will be too high.

We will investigate the relation between these tolerances and the absolute computation error. We considered that an approximation of the real solution is obtained after a high number of full Newton-Raphson iterations (100), therefore we can compute an approximation of the real maximum absolute error after each iteration as:

$$Err^i = \left\| U_a^i - U_a^{100} \right\|_\infty \tag{17}$$

By plotting the absolute error given by (17) against the number of iterations for the two analyzed problems (Fig. 5) we can see that in both cases the solution is converging asymptotically, but the convergence is faster for the cylinder compression problem than for the ellipsoid deformation problem. For the last iterations there is little improvement in the solution accuracy, therefore the solution approximation must be very close to the real solution.

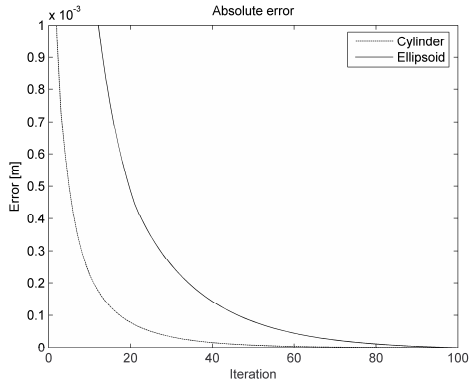


Fig. 5. Absolute error variation versus the number of iterations

We can now analyze the variation of the quantities ε^i_D, ε^i_F and ε^i_E with the absolute error for the two problems when the Newton-Raphson iterations are used – see Fig. 6-7. From these graphs we can see that there is a non-linear relation between the convergence tolerances (14-16) and the absolute error obtained and that for different problems there is a big difference between the values of the convergence tolerances that ensures the same absolute error. This makes the selection of the convergence tolerances very difficult.

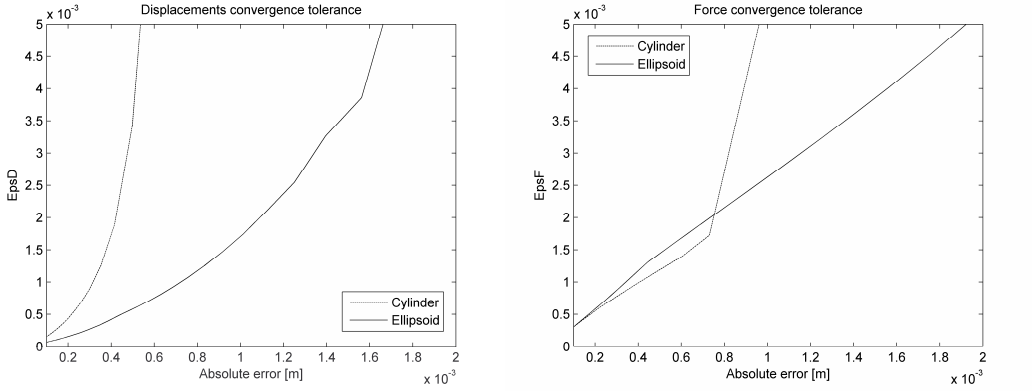


Fig. 6. Displacements and force convergence tolerance versus absolute error

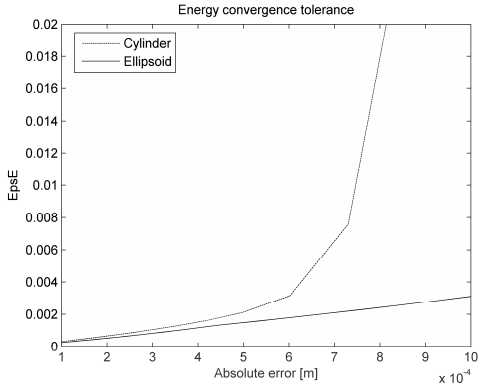


Fig. 7. Energy convergence tolerance versus absolute error

In [34] a proposal is made to use a convergence criteria based on a convergence factor. Starting from their proposal, we define a convergence factor as:

$$q^i = \frac{\left\| U_a^i - U_a^{i-1} \right\|_\infty}{\left\| U_a^{i-1} - U_a^{i-2} \right\|_\infty} \tag{18}$$

It is easy to show, using the properties of norms, that if the iteration is converging asymptotically (q^i<1) and the convergence factor for the future iterations does not exceed the current convergence factor, then

$$\left\| U_a^i - U_a^* \right\|_\infty \le \frac{q^i}{1-q^i} \left\| U_a^i - U_a^{i-1} \right\|_\infty \tag{19}$$

where $U^*_a = U^\infty_a$ is the real solution. However, by studying the variation of the convergence factor (Fig. 8) we can see that the conditions are not met, because the convergence factor is not decreasing, but very slowly increasing with the number of iterations, and therefore (19) does not hold. By comparing the right hand side of (19) with the absolute error (17) (Fig. 9.a), we can see that indeed (19) under-estimates the absolute error. Therefore, we propose to apply a correction to the error estimation (19), obtaining the corrected error estimation as:

$$EstErr^i = \frac{1}{q^i(1-q^i)} \left\| U_a^i - U_a^{i-1} \right\|_\infty \tag{20}$$

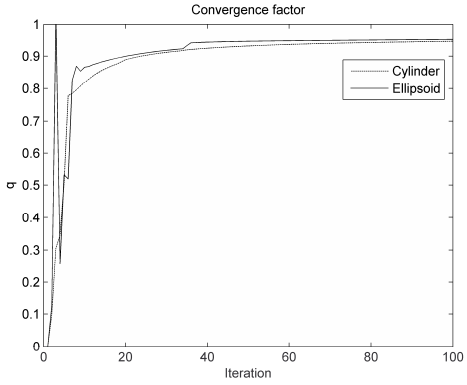
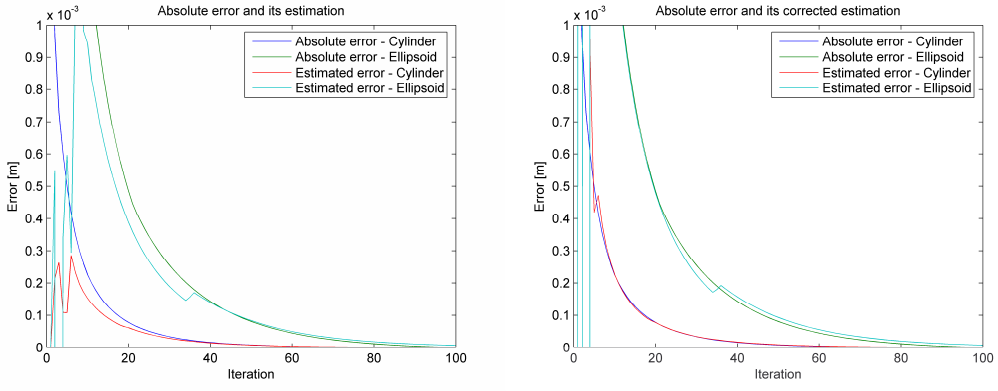


Fig. 8. Convergence factor variation



a) b)

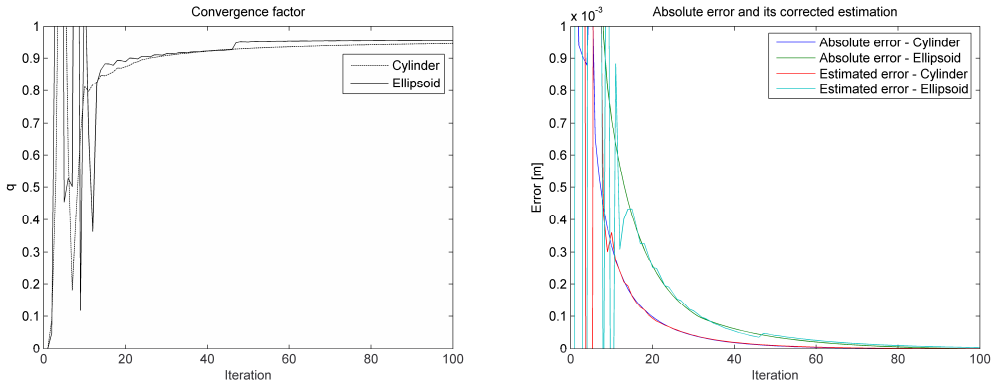
Fig. 9. Error estimation (a) and corrected error estimation (b)

Fig. 9.b shows that (20) is a very good estimate of the absolute error for both problems when the Newton-Raphson iterations are used (after the convergence factor stabilizes). Therefore, a useful convergence criterion is:

$$EstErr^i \leq AbsErr \tag{21}$$

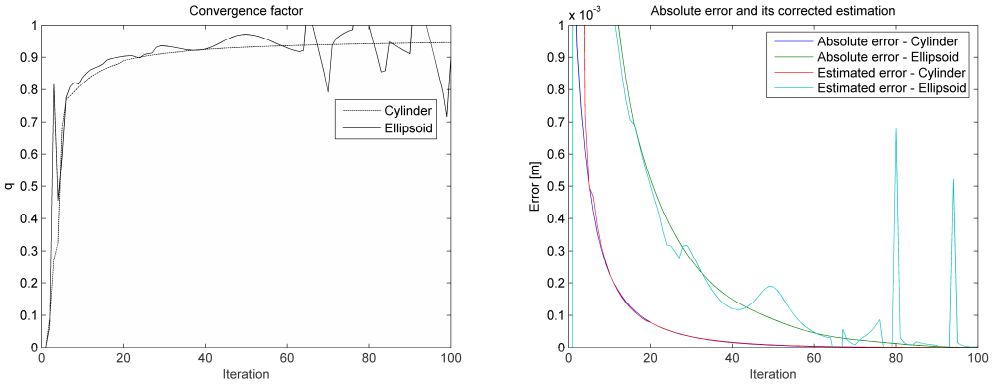
where *AbsErr* is the maximum error allowed for the computation results. In case of medical imaging, the image resolution obtained using MRI have the accuracy of around 1 mm, therefore we can ask that the computation accuracy (*AbsErr*) to be in the range of [0.1, 1] mm.

We can study now whether (21) can be used as a convergence criterion in case of modified and composite Newton methods. For the modified Newton method, the variation of the convergence factor (18) with the number of iterations is presented in Fig. 10.a and the corrected error estimate (20) is compared with the actual error in Fig. 10.b. The tangent stiffness matrix is updated every 4 iterations for the cylinder compression problem but the analysis did not converge in case of the ellipsoid deformation problem. To ensure the convergence we had to reduce the maximum displacement size to 0.02 m and to update the tangent stiffness matrix every 3 iterations. We can see that after the value of the convergence factor stabilizes, the corrected error estimate is very close to the actual error. Nevertheless, it takes more iterations for the convergence factor to stabilize and afterwards its variation is not as smooth as in case of full Newton-Raphson method.



a) b)

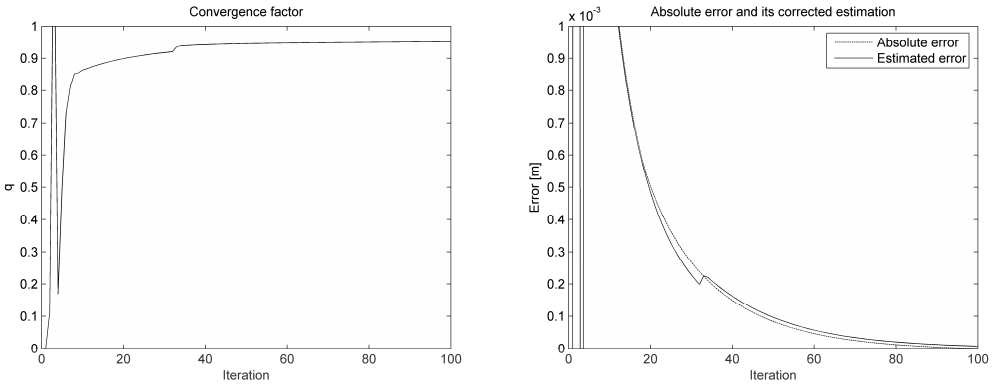
Fig. 10. Convergence factor variation (a) and corrected error estimation (b) for the modified Newton method



a) b)

Fig. 11. Convergence factor variation (a) and corrected error estimation (b) for the composite Newton-JCG and Newton-SSORSI methods

In case of the composite Newton methods, when solving the selected problems using the iterative methods and the number of iterations identified in 7.1 we obtain the variation of the convergence factor presented in Fig. 11.a. It seems that the convergence factor does not stabilize in case of the ellipsoid deformation, when SSORSI method is used. This also leads to a bad approximation of the error, as presented in Fig. 11.b. Therefore we tried to change the iterative method used, and selected the next best iterative method, JCG, with 250 iterations (see 7.1). In this case the convergence factor has the expected variation and the error estimate can be done using (21) – Fig. 12.



a) b)

Fig. 12. Convergence factor variation (a) and corrected error estimation (b) for the composite Newton-JCG method (ellipsoid deformation)

Based on the previous analysis, we propose the usage of two convergence criteria when the static solution of a finite element problem is computed. The first criterion is looser, and must ensure that the solution remains convergent between 2 load steps – we chose the residual forces convergence criterion (15). The second criterion is based on (21) and ensures that the provided solution is within required error limits. This criterion only applies on the last load step, after the first convergence condition has been met (this should also ensure that the convergence factor has stabilized its value, fact that must be checked in the code).

By using this approach we allow the solution to advance faster from one load step to the next, but do not stop the iterations until the required precision is obtained.

3.6.3. Computation time

For both the analyzed problems we choose the residual forces convergence tolerance ε_{FLim}=0.005 and we require that the maximum computation error in any direction should be less than 0.5 mm. That means that the absolute error limit in (21) would be (in the worse case scenario, as this is a limit of error along the axes):

$$AbsErr = \frac{0.0005}{\sqrt{3}} = 2.88e-4 \tag{22}$$

When the modified Newton method is used, the tangent stiffness matrix is updated every 4 iterations. For the composite Newton methods the JCG iterative method is used, with 250 iterations for the ellipsoid deformation problem and 150 iterations for the cylinder compression problem. Line search is used with maximum 5 search iterations.

The problem properties are presented in table 1.

Table 1. Problem properties

Mesh	**Elements**	**Nodes**	**Equations**
Cylinder	6000	6741	18297
Ellipsoid	2200	2535	7260

The computation time is presented in table 2. The maximum error in any direction is computed after the last iteration, n, as:

$$Error = \max_i \left\| P_i^n - P_i^{100} \right\|_2 \tag{23}$$

where P_i^n is the position of node i after iteration n. The maximum error along the axis – *Err* - is given by (17).

Table 2. Computation time and maximum error

Problem	**Method**	**Itera-tions**	**Time [s]**	**Err [mm]**	**Error [mm]**
Cylinder compression	Full Newton-Raphson	9	66.2	0.26	0.27
	Modified Newton	11	30.9	0.27	0.28
	Composite Newton-JCG	9	21.3	0.26	0.27
Ellipsoid deformation	Full Newton-Raphson	27	57.0	0.31	0.35
	Modified Newton	-	-	-	-
	Composite Newton-JCG	27	28.7	0.32	0.34
	Composite Newton-JCG – 150 JCG iterations	27	22.5	0.31	0.35

In case of the ellipsoid deformation problem the modified Newton method did not converge. We also used the composite Newton-JCG method with only 150 JCG iterations to solve the ellipsoid deformation problem.

3.6.4. Conclusions

The composite methods converged in the same number of iterations as the full Newton-Raphson methods, but are much more efficient from the computational point of view, reducing the computation time up to 3 times for the studied problems.

The modified Newton methods need more iterations than the full Newton methods but can lead to a decrease in the computation time, although they are not as good as the composite methods. Also the convergence behavior is not very good, as sometimes these methods diverge while the others converge.

The composite Newton-JCG method is very robust, its convergence remaining the same for the ellipsoid deformation problem when the number of JCG iterations was reduced from 250 to 150.

The proposed convergence criteria proved to be very efficient, as the obtained error was always below the imposed limit (0.5 mm).

To prove the good performances of the direct solver we used in our implementation of the full Newton-Raphson method (MUMPS), we computed the solution to the cylinder compression problem using static analysis with the direct solver in the commercial finite element program ABAQUS [7]. This program performed 4 iterations in 34 seconds, therefore having a computation speed comparable with the speed of our solver (8.5 s/iteration compared to 7.3s/iteration).

4. Explicit analysis algorithms

The integration of equilibrium equations in time domain can be done using either implicit or explicit methods [13, 57, 58]. The most commonly used implicit integration methods, such as the Newmarks' constant acceleration method, are unconditionally stable. This implies that their time step is limited only by the convergence/accuracy considerations. However, the implicit methods require solution of set of non-linear algebraic equations at each time step. Furthermore, iterations need to be performed for each time step of implicit integration to control the error and prevent divergence. Therefore, the number of numerical operations per each time step can be three orders of magnitude larger than for explicit integration [57].

On the other hand, in explicit methods, such as a central difference method, treatment of non-linearities is very straightforward and no iterations are required. By using a lumped (diagonalised) mass matrix, the equations of motion can be decoupled and no system of equations must be solved. Computations are done at the element level eliminating the need for assembling the stiffness matrix of the entire model. Thus, computational cost of each time step and internal memory requirements are substantially smaller for explicit than implicit integration. There is no need for iterations anywhere in the algorithm. These features make explicit integration suitable for real time applications.

However, the explicit methods are only conditionally stable. Normally a severe restriction on the time step size has to be included in order to receive satisfactory simulation results. Stiffness of soft tissue is very low [14, 59, 60]: e.g. stiffness of brain is about eight orders of magnitude lower than that of steel. Since the maximum time step allowed for stability is (roughly speaking) inversely proportional to the square root of Young's modulus divided by the mass density, it is possible to conduct simulations of brain deformation with much longer time steps than in typical dynamic simulations in engineering. A detailed description of the Total Lagrange Explicit Dynamics [TLED] algorithm is presented in [33].

For the cylinder compression problem presented above, the TLED algorithm can perform 1000 time steps in less than 6 seconds. This makes it a good option for real time surgical simulation.

5. Conclusions

In this report we presented a suite of finite element algorithms that can be used for accurate and fast computation of soft tissue deformation for surgical simulation. The basic concept between these algorithms is the usage of the Total Lagrangian formulation for solving finite element problems. The presented algorithms cover issues related to time integration, hourglassing and volumetric locking. We use fully nonlinear formulation, accounting for large deformations, rigid body motions and material non-linearities.

A new tetrahedral element formulation based on the average nodal Jacobian was developed. This formulation uses only kinematic variables for controlling the volumetric locking, and therefore the usage of different materials and the implementation in an existing finite element code can be made without difficulties.

A very efficient hourglass control implementation is proposed for the under-integrated hexahedral element. Having only one integration point, this element is very inexpensive from the computational point of view, being a perfect candidate for real time surgical simulations. Using this type of element and the improved tetrahedral element for simulations using mixed meshes is a step towards complete automated patient specific surgical simulation.

We proposed to use composite Newton methods for solving the nonlinear system of equations obtained from a finite element model that describes the deformation of a soft organ. A method was described for choosing the best iterative method for a given problem. From the different available implementations of iterative methods for solving linear systems of equations, the JCG method offered the best performances in terms of computation time and convergence properties for the studied examples.

Based on the convergence properties of the full Newton-Raphson method and composite Newton-JCG method, a very good approximation of the maximum computation error can be computed using a convergence factor. This error approximation can then be integrated into a convergence criterion in order to ensure the desired accuracy of the computations. We proposed the usage of two convergence criteria: one that can ensure the convergence between different load steps and one that ensures the accuracy of the results. The last criterion is applied only at the end of the last load step. This ensures that no time is lost computing intermediate results to higher that needed accuracy.

The composite Newton-JCG method decreased the computation time more than 3 times for the studied problems. It converged in the same number of iterations as the full Newton

method, with similar accuracy of the computed results. It also proved very robust with respect to the configured number of JCG iterations.

Explicit time integration is the preferred method for performing real time simulations. The treatment of non-linearities is straightforward, without the need of any iteration. Even if the method is only conditionally stable, the material properties of biological tissue make possible the usage of much larger time steps compared with other engineering applications.

The simulation examples confirm the speed of the TLED algorithm. We could compute reaction forces at frequencies higher than 500 kHz for a mesh having more than 2000 hexahedral elements on a simple PC workstation.

References

1. Oden, J.T., Belytschko, T., Babuska, I., and Hughes, T.J.R. Research directions in computational mechanics. *Comput. Methods Appl. Mech. Engrg.* 2003; 192: 913-922
2. Carey, G.F. A Unified Approach to Three Finite Element Theories for Geometric Nonlinearity. *Journal of Computer Methods in Applied Mechanics and Engineering* 1974; 4(1): 69-79
3. Martin, H.C. and Carey, G.F. *Introduction to Finite Element Analysis: Theory and Application*. McGraw-Hill Book Co.: New York, 1973
4. Oden, J.T. and Carey, G.F. *Finite Elements: Special Problems in Solid Mechanics*. Prentice-Hall, 1983
5. Belytschko, T. An Overview of Semidiscretization and Time Integration Procedures, in Computational Methods for Transient Analysis,Editor:T. Belytschko and T.J.R. Hughes (1983) North-Holland: Amsterdam. p. 1-66
6. http://www.ansys.com/services/documentation, ANSYS documentation, Accessed: 18/10/2005
7. ABAQUS. *ABAQUS Theory Manual, Version 5.8*. Hibbitt, Karlsson & Sorensen, Inc., 1998
8. www.adina.com, ADINA home page, Accessed: 22/11/2005
9. Hallquist, J.O. *LS-DYNA Theory Manual*. Livermore Software Technology Corporation: Livermore, California 94551, 2005
10. Zhuang, Y. and Canny, J. Real-time Simulation of Physically Realistic Global Deformation. *IEEE Vis'99* (1999) San Francisco, California
11. Picinbono, G., Delingette, H., and Ayache, N. Non-linear anisotropic elasticity for real-time surgery simulation. *Graphical Models* 2003; 65: 305-321
12. Fung, Y.C. *Biomechanics. Mechanical Properties of Living Tissues*. Second ed. Springer-Verlag: New York, 1993
13. Bathe, K.-J. *Finite Element Procedures*. Prentice-Hall: New Jersey, 1996
14. Miller, K. *Biomechanics of Brain for Computer Integrated Surgery*. Publishing House of Warsaw University of Technology: Warsaw, 2002
15. Owen, S.J. A Survey of Unstructured Mesh Generation Technology. *7th International Meshing Roundtable* (1998) Dearborn, Michigan, USA
16. Viceconti, M. and Taddei, F. Automatic generation of finite element meshes from computed tomography data. *Critical Reviews in Biomedical Engineering* 2003; 31(1): 27-72
17. Owen, S.J. Hex-dominant mesh generation using 3D constrained triangulation. *Computer-Aided Design* 2001; 33: 211-220
18. Ferrant, M., Macq, B., Nabavi, A., and Warfield, S.K. Deformable Modeling for Characterizing Biomedical Shape Changes. *Discrete Geometry for Computer Imagery: 9th International Conference* (2000) Uppsala, Sweden: Springer-Verlag GmbH
19. Ferrant, M., Nabavi, A., Macq, B., Black, P.M., Jolesz, F.A., Kikinis, R., and Warfield, S.K. Serial registration of intraoperative MR images of the brain. *Medical Image Analysis* 2002; 6(4): 337-359
20. Clatz, O., Delingette, H., Bardinet, E., Dormont, D., and Ayache, N. Patient Specific Biomechanical Model of the Brain: Application to Parkinson's disease procedure. *International Symposium on Surgery Simulation and Soft Tissue Modeling (IS4TM'03)* (2003) Juan-les-Pins, France: Springer-Verlag
21. Warfield, S.K., Talos, F., Tei, A., Bharatha, A., Nabavi, A., Ferrant, M., Black, P.M., Jolesz, F.A., and Kikinis, R. Real-time registration of volumetric brain MRI by

biomechanical simulation of deformation during image guided surgery. *Computing and Visualization in Science* 2002; 5: 3-11
22. Flanagan, D.P. and Belytschko, T. A uniform strain hexahedron and quadrilateral with orthogonal hourglass control. *International Journal for Numerical Methods in Engineering* 1981; 17: 679-706
23. *ABAQUS/Explicit User's Manual Version 6.1*. Hibbitt, Karlsson & Sorensen, Inc.: Pawtucket, 2000
24. Liu, W.K., Ong, J.S.-J., and Uras, R.A. Finite element stabilization matrices-a unification approach. *Computer Methods in Applied Mechanics and Engineering* 1985; 53(1): 13-46
25. Puso, M.A. A highly efficient enhanced assumed strain physically stabilized hexahedral element. *International Journal for Numerical Methods in Engineering* 2000; 49: 1029-1064
26. Belytschko, T. and Bindeman, L.P. Assumed strain stabilization of the eight node hexahedral element. *Computer Methods in Applied Mechanics and Engineering* 1993; 105: 225-260
27. Reese, S. and Wriggers, P. A stabilization technique to avoid hourglassing in finite elasticity. *International Journal for Numerical Methods in Engineering* 2000; 48: 79-109
28. Joldes, G.R., Wittek, A., and Miller, K. Efficient Hourglass Control Implementation for the Uniform Strain Hexahedron Using the Total Lagrangian Formulation. *Communications in Numerical Methods in Engineering* 2007; Submitted, under review
29. Bonet, J. and Burton, A.J. A simple averaged nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. *Communications in Numerical Methods in Engineering* 1998; 14: 437-449
30. Bonet, J., Marriott, H., and Hassan, O. An averaged nodal deformation gradient linear tetrahedral element for large strain explicit dynamic applications. *Communications in Numerical Methods in Engineering* 2001; 17: 551-561
31. Zienkiewicz, O.C., Rojek, J., Taylor, R.L., and Pastor, M. Triangles and Tetrahedra in Explicit Dynamic Codes for Solids. *International Journal for Numerical Methods in Engineering* 1998; 43: 565-583
32. Joldes, G.R., Wittek, A., and Miller, K. Improved Linear Tetrahedral Element for Surgical Simulation. *Computational Biomechanics for Medicine - MICCAI 2006 Workshop Proceedings* (2006) Copenhagen, Denmark
33. Miller, K., Joldes, G.R., Lance, D., and Wittek, A. Total Lagrangian Explicit Dynamics Finite Element Algorithm for Computing Soft Tissue Deformation. *Communications in Numerical Methods in Engineering* 2007; 23: 121-134
34. Bathe, K.-J. and Cimento, A.P. Some practical procedures for solution of nonlinear finite element equations. *Computer Methods in Applied Mechanics and Engineering* 1980; 22: 59-85
35. Zienkiewicz, O.C. and Taylor, R.L. *The Finite Element Method - Solid Mechanics*. 5 ed Vol. 2. Butterworth-Heinemann: Oxford, 2000
36. Ortega, J.M. and Rheinboldt, W.C. *Iterative solution of nonlinear equations in several variables*. Academic Press: New York and London, 1970
37. Dahlquist, G. and Bjork, A. *Numerical Methods*. Prentice-Hall, Inc: Englewood Cliffs, 1974
38. Barnard, S.T. and Grote, M.J. A Block Version of the SPAI Preconditioner. *Proc. 9th SIAM Conf. on Parallel Process. for Sci. Comp* 1999;
39. Hageman, L. and Young, D. *Applied Iterative Methods*. Academic Press: New York, 1981

40. Sawada, S. and Kudo, H. Globally convergent Newton-SOR method for statistical imagereconstruction. *Nuclear Science Symposium Conference Record, 2000 IEEE* (2000) Lyon, France
41. Mittelmann, H.D. On the Efficient Solution of Nonlinear Finite Element Equations. *Numerische Mathematik* 1980; 35: 277-291
42. Fletcher, R. *Practical Methods of Optimization - Unconstrained Optimization* Vol. 1. John Willey & Sons, Ltd.: New York, 1980
43. Dennis, J.E. and Schnabel, R.B. *Numerical Methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Inc.: Englewood Cliffs, New Jersey, 1983
44. Brent, R. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, 1973
45. http://mumps.enseeiht.fr/index.html, MUMPS home page, Accessed: 30 Jan 2007
46. Gould, N.I.M., Hu, Y., and Scott, J.A.,A Numerical Evaluation of Sparse Direct Solvers for the Solution of Large Sparse, Symmetric Linear Systems of Equations, Editor: Editor (2005) Council for the Central Laboratory of the Research Councils
47. Cuthill, E. and McKee, J. Reducing the bandwidth of sparse symmetric matrices. *Proceedings of the 1969 24th national conference* (1969) New York: ACM Press
48. Gibbs, N.E., Poole, W.G., and Stockemeyer, P.K. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal of Numerical Analysis* 1976; 13(2): 236-250
49. http://www.math.temple.edu/~daffi/software/rcm/, Accessed: 30 Jan 2007
50. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and Vorst, H.V.d. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 2 ed. SIAM: Philadelphia, PA, 1994
51. http://math.nist.gov/iml++/, IML++ home page, Accessed: 30 Jan 2007
52. Dongarra, J., Lumsdaine, A., Pozo, R., and Remington, K. A Sparse Matrix Library in C++ for High Performance Architectures. *Proceedings of the Second Object Oriented Numerics Conference* (1992)
53. http://math.nist.gov/sparselib++/, SparseLib++ home page, Accessed: 30 Jan 3007
54. Kincaid, D. and Young, D. The ITPACK Package for Large Sparse Linear Systems, in Elliptic Problem Solvers,Editor:M. Schultz (1981) Academic Press: New York. p. 163-185
55. http://www.netlib.org/itpack/, ITPACK home page, Accessed: 30 Jan 2007
56. Miller, K., Chinzei, K., Orssengo, G., and Bednarz, P. Mechanical properties of brain tissue in-vivo: experiment and computer simulation. *Journal of Biomechanics* 2000; 33: 1369-1376
57. Belytschko, T. A survey of numerical methods and computer programs for dynamic structural analysis. *Nuclear Engineering and Design* 1976; 37: 23-34
58. Crisfield, M.A. Non-linear dynamics, in Non-linear Finite Element Analysis of Solids and Structures (1998) John Wiley & Sons: Chichester. p. 447-489
59. Miller, K. and Chinzei, K. Constitutive modelling of brain tissue; Experiment and Theory. *Journal of Biomechanics* 1997; 30(11/12): 1115-1121
60. Miller, K. and Chinzei, K. Mechanical properties of brain tissue in tension. *Journal of Biomechanics* 2002; 35: 483-490