Chapter 9
Algorithms for Computational Biomechanics of the Brain

A. Wittek, G. Joldes, and K. Miller

9.1 Introduction

Modeling of the brain responses due to injury-causing transients and surgery is a problem of continuum mechanics that involves irregular geometry, complex loading and boundary conditions, non-linear materials, and large deformations (see Chaps. 5 and 6). Finding a solution for such a problem requires computational algorithms of non-linear continuum mechanics.

As stated in Chap. 5, modeling for brain injury simulation has been driven by the idea that numerical surrogates of the human brain can be used in the design of countermeasures mitigating the traumatic brain injury. Such modeling has been done with significant contribution and involvement of the automotive manufacturers [1] and participation of organizations responsible for traffic safety (e.g. National Highway Traffic Safety Administration NHTSA) [2]. Because of these industrial links, modeling of the brain for injury simulation has been dominated by the explicit dynamics (i.e. utilizing explicit integration in time domain) non-linear finite element algorithms available in commercial finite element codes, such as LS-DYNA [3], PAM-SAFE [4], RADIOSS [5], that are routinely used by the automotive industry.

In computational biomechanics for medicine, on the other hand, significant research effort has been directed into the development of specialized algorithms that can provide the results within the real-time constraints of surgery. For instance, great interest was given to mass-spring method [6, 7] in which the analyzed continuum is modeled as a discrete system of nodes (where the mass is concentrated) and springs. Due to its simplicity and low computational complexity, this method was applied in simulation software for virtual reality training systems for

A. Wittek (✉)
Intelligent Systems for Medicine Laboratory, School of Mechanical and Chemical Engineering, The University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia
e-mail: adwit@mech.uwa.edu.au

K. Miller (ed.), *Biomechanics of the Brain*, Biological and Medical Physics, Biomedical Engineering, DOI 10.1007/978-1-4419-9997-9_9,

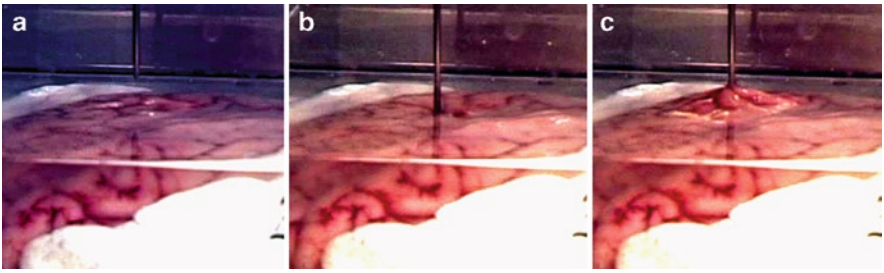

Fig. 9.1 Swine brain deformation during needle insertion. (**a**) Before insertion; (**b**) during the insertion (note the deformation in the insertion area); (**c**) during the needle removal. The experiments were conducted at the laboratory of the Surgical Assist Technology Group, Group, Institute for Human Science and Biomedical Engineering, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Ibaraki, Japan. For the experiment description, see [31]

minimally invasive surgery [8]. However, the behavior of the mass-spring models strongly depends on the topology of the spring network and spring parameters are difficult to identify and express in terms of the soft tissue constitutive parameters (such as Young's modulus and Poisson's ratio) that are used in continuum mechanics [9]. Therefore, in recent years, more interest has been given to the finite element method [10] that utilizes the principles of continuum mechanics and does not suffer from the limitations of the mass-spring method.

Traditionally, real-time computations for biomechanics for medicine relied on linear finite element algorithms that assume infinitesimally small deformations [9, 11–14]. However, this assumption is not satisfied in surgical procedures where large deformations of the organ undergoing surgery occur. Examples include the brain deformations due to craniotomy (referred to in the literature as brain shift [15]) (see Fig. 6.1 in Chap. 6) and needle insertion (Fig. 9.1 above). Therefore, we focus on the algorithms that utilize fully non-linear (i.e. accounting for finite deformations and non-linear stress–strain relationships of soft tissues) formulation of solid mechanics that can be applied to any situation. In such a formulation, the current volume and surface of the modeled body organ (over which the integration of equations of continuum mechanics is to be conducted) are unknown. They are part of the solution rather than input data (Fig. 9.2). The literature indicates that taking into account geometric non-linearity (through finite deformation formulation of the equations of continuum mechanics) is needed to ensure accuracy of prediction of soft organ deformations even for applications that do not involve very large strains (e.g. brain shift) [16, 17].

In the subsequent sections of this chapter, we discuss the following topics:

- Section 9.2: Non-linear explicit dynamics finite element algorithms implemented in commercial finite element codes applied to modeling brain injury biomechanics.
- Section 9.3: A specialized non-linear finite element algorithm for surgery simulation that utilizes explicit integration in time domain and Total Lagrangian incremental formulation of continuum mechanics.
- Section 9.4: A specialized non-linear finite element algorithm that utilizes Dynamic Relaxation and Total Lagrangian formulation for computation of steady-state brain deformation within the real-time constraints of image-guided neurosurgery.

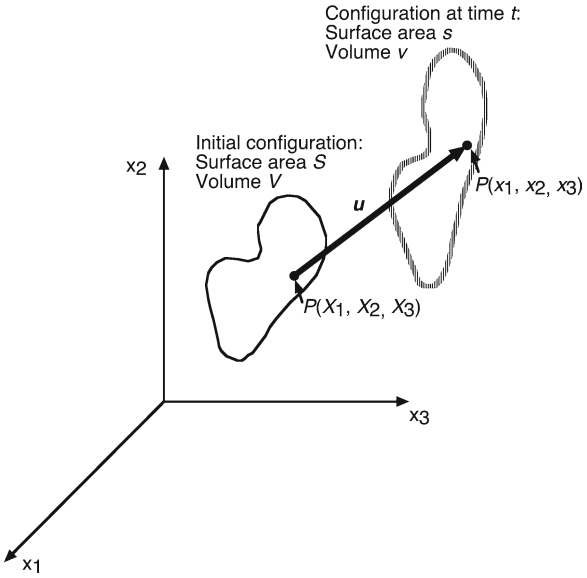

Fig. 9.2 Computational biomechanics as a non-linear problem of continuum mechanics. During impact and surgery, human body organs undergo large displacements (composed of rigid body motions and local deformations). Consequently, the equations of continuum mechanics governing the organ behavior need to be integrated over the current surface s and volume v

- Section 9.5: Element formulation for the specialized algorithms for surgery simulation and neurosurgery modeling. This includes non-locking tetrahedral element and efficient hourglass control for hexahedral element.
- Section 9.6: An efficient sliding contact algorithm for modeling of brain–skull interaction for image-guided neurosurgery.
- Section 9.7: Meshless algorithms that utilize a computational grid in the form of a cloud of points as one possible remedy for the limitations of finite element algorithms (including time-consuming generation of patient-specific computation grids as well as deterioration of the computation accuracy and instability due to very large strains induced by surgery).
- Section 9.8: Implementation of the specialized non-linear finite element algorithms for neurosurgery modeling on graphics processing units (GPUs) for real-time solution of the brain models for computer-assisted neurosurgery.
- Section 9.9: Algorithm verification.

9.2 Algorithms for Injury Simulation

Impact/injury biomechanics of the brain deals with events of very short duration (hundreds of milliseconds) in which the head is subjected to transient loading due to either direct impact or rapid acceleration that results in large deformation (or even

mechanical damage) of the brain tissue. As indicated in [10, 18, 19], non-linear finite element procedures with explicit integration in time domain outperform other algorithms in modeling of three-dimensional continua in such events. Therefore, they have been a preferable choice in brain injury biomechanics (see also Chap. 5) and implemented in numerous finite elements codes (such as e.g. LS-DYNA, PAM-CRASH, ABAQUS, RADIOSS) industrially applied in impact injury simulation.

In impact injury simulation and other transient dynamics problems, the global system of finite element equations to be solved at each time step is:

$$\mathbf{M}\ddot{\mathbf{u}}_{n+1} + \mathbf{K}(\mathbf{u}_{n+1}) \cdot \mathbf{u}_{n+1} = \mathbf{R}_{n+1}, \tag{9.1}$$

where $\mathbf{u}$ is a vector of nodal displacements, $\mathbf{M}$ is a mass matrix, $\mathbf{K}$ is a stiffness matrix non-linearly dependent on the deformation (because of the non-linear material model), and $\mathbf{R}$ is a vector of nodal (active) forces. The product of the stiffness matrix and nodal displacements vector gives the nodal reaction forces vector $\mathbf{F}$. In explicit dynamics finite element procedures, the accelerations determined from equation of motion (9.1) are integrated to calculate the displacements using difference methods. Although many difference methods exist [10], the central difference method is the most commonly used due to its efficiency [10, 18]:

$$u_{n+1} = u_n + \Delta t_{n+1}\dot{u}_n + \frac{1}{2}\Delta t_{n+1}^2 \ddot{u}_n, \tag{9.2}$$

$$\dot{u}_{n+1} = \dot{u}_n + \frac{1}{2}\Delta t_{n+1}(\ddot{u}_{n+1} + \ddot{u}_n). \tag{9.3}$$

Combining the central difference method given by (9.2) to (9.3) with the global system of finite element (9.1) yields the following formula for the vector of nodal displacements $\mathbf{u}_{n+1}$ at integration step $n+1$:

$$\mathbf{M}\mathbf{u}_{n+1} = \Delta t^2 \left(\mathbf{R}_n - \sum_i \mathbf{F}_n^{(i)} \right) - \mathbf{M}(\mathbf{u}_{n-1} - 2\mathbf{u}_n). \tag{9.4}$$

Formally (9.4) represents a system of equations. This system can be decoupled by using a lumped mass matrix $\mathbf{M}$. For lumped mass matrices, all non-diagonal components equal zero (i.e. the mass distribution is discretized by placing the particle masses at the nodes of an element). For such matrices, (9.4) becomes an explicit formula for the unknown nodal displacements $\mathbf{u}_{n+1}$:

$$\mathbf{u}_{n+1}^j = \Delta t^2 \left(\mathbf{R}_n^j - \sum_i \mathbf{F}_n^{(i)j} \right) / m_{jj} - (\mathbf{u}_{n-1}^j - 2\mathbf{u}_n^j), \tag{9.5}$$

where $\mathbf{u}_{n+1}^j$ is a vector of nodal displacements at node j at intergation step $n+1$, m_{jj} is the component of the lumped mass matrix corresponding to node j (i.e. mass lumped to node j), $\mathbf{R}_n^j$ is the vector of external forces applied to node j, $\sum_i \mathbf{F}_n^{(i)j}$ is

the vector of nodal reaction forces at node j (sum of the contribution of the elements i to which the node belongs), and Δt is the integration step.

In (9.5), the computations are done at element level eliminating the need for assembling the stiffness matrix of the entire analyzed continuum. The mechanical properties of the analyzed continuum are accounted for in the constitutive model and included in the calculation of nodal reaction forces **F**. Thus, computational cost of each time step and internal memory requirements are very low. It is worth noting that there is no need for iterations anywhere in the algorithm summarized in (9.5), even for non-linear problems. This implies the following advantages of non-linear finite element procedures utilizing explicit integration in time domain and mass lumping:

- Straightforward treatment of non-linearities
- No iterations required for a time step
- No need to solve a system of equations
- Low computational cost for each time step
- Low internal memory requirements

However, explicit methods are only conditionally stable. A restriction (Courant criterion [20]) on the time step size has to be included in order to obtain stable simulation results. The time step (referred to as a critical time step) that ensures the computation stability is equal to the smallest characteristic length of an element in the mesh divided by the dilatational (acoustic) wave speed [18, 20, 21]:

$$\Delta t \leq \frac{L_e}{c}, \tag{9.6}$$

where L_e is the smallest characteristic element length in the model and c is the dilatational (acoustic) wave speed. The formulae for calculating L_e for various commonly used elements are given in [22].

Equation (9.6) is equivalent to setting the condition that the time step cannot be longer than the travel time of the wave across the smallest element in the mesh [18]. Based on [22], the acoustic wave speed can be expressed as

$$c = \sqrt{\frac{E}{\rho} \frac{(1-\nu)}{(1+\nu)(1-2\nu)}}, \tag{9.7}$$

where E is the Young's modulus, ρ is the density, and ν is the Poisson's ratio.

It should be noted that the stability limit given in (9.6) has been derived for linear problems. However, according to [18], there is considerable empirical evidence that it is also valid for non-linear problems.

Equations (9.6) and (9.7) imply that the critical time step can be increased by increasing the density (and hence the mass) of the smallest elements (as determined by the characteristic length L_e) in the mesh. This process is referred to as mass scaling [3, 23]. Moderate mass scaling does not significantly change the responses of the analyzed continuum and is regarded as a powerful method for decreasing the

computation time [24]. However, it is also acknowledged that too severe scaling can introduce non-physical inertia effects [24]. Formal guidelines for determining permissible mass scaling limits for injury simulation have not yet been formulated and the quantitative information about the scaling that is used is rarely provided in the biomechanical literature. Majumder et al. [25] reported a significant reduction in computation time for scaling resulting in the total model mass increase as low as 0.016%. ABAQUS finite element solver manual [26] recommends that, after applying mass scaling, the kinetic energy of the system should not exceed 5% of the total energy over most of the analysis.

Stress calculation for obtaining the nodal forces **F** is the major computation cost of the explicit dynamic finite element procedures summarized in (9.4) and (9.5). This implies that the complexity of the elements and the number of integration points per element are the key factors determining the number of computations in these procedures. Therefore, as mentioned in Chap. 5, in injury simulation utilizing non-linear finite element procedures with explicit integration in time domain, 8-noded hexahedron [27, 28] and 4-noded tetrahedron [27, 28] with linear shape functions and one integration (Gauss) point are the most commonly used elements.

The 8-noded hexahedron with one integration point is an under-integrated element (or low-order Gauss quadrature element), i.e. an element for which the stiffness matrix rank is lower than the number of element's degrees of freedom minus the number of rigid body modes [29]. Under-integrated elements exhibit an instability known as hourglassing or zero-energy modes, i.e. there are nodal displacement vectors which produce no strain energy, but do not define a rigid body motion [27, 29].

The 4-noded tetrahedral elements with linear shape functions and one Gauss point do not suffer from hourglassing. However, for incompressible (or nearly incompressible) continua, such as soft tissues, 4-noded tetrahedral elements exhibit artificial stiffening known as volumetric locking [20, 30]. A more detailed discussion on algorithms for hourglass control and volumetric locking reduction is provided in Sect. 9.3.

9.3 Algorithms for Surgery Simulation

As discussed in Chap. 6, surgical simulation systems are required to provide visual and haptic feedback to a surgeon or trainee. Such systems must provide a time accurate prediction of the deformation field within an organ and interaction force between the surgical tool and the tissue at frequencies of at least 500 Hz. From the perspective of continuum mechanics, such prediction requires solving the problem involving large deformations, non-linear constitutive properties and non-linear boundary conditions within the very strict time constraints of haptic feedback.

For relatively slow varying loads, such as those that occur due to interactions between tissue and surgical tool, non-linear finite element algorithms utilizing implicit integration in time domain are traditionally recommended in the literature for solving non-linear problems of solid mechanics [10]. Such procedures rely on solving systems of algebraic equations and require computationally expensive iterations. In contrast, the algorithms for injury simulation lead to an explicit formula for unknown nodal

displacements (9.5). For such algorithms, the number of operations per integration step is typically three orders of magnitude smaller than for the ones relying on implicit integration [18]. Despite the fact that surgical simulations involve phenomena with durations that are orders of magnitude longer than those that are of interest in injury simulation, the restrictions on the time step size (Courant criterion) required for solution stability (9.6) do not compromise efficiency of explicit time integration in such simulations. This is because the acoustic wave speed is proportional to the square root of the analyzed continuum Young's modulus (9.7) which is very low (under 10^4 Pa) for brain tissue. For instance, in the simulation of needle insertion into brain conducted using non-linear explicit dynamics finite element procedures reported in [31], the time step was over 1.5×10^{-2} ms. In contrast, typical engineering applications, such as metal forming, use integration steps that are of an order of 10^{-5} ms [32].

In commercial finite element codes utilizing explicit time integration, the calculated variables (such as displacement, strain and stress) are incremented by referring them to the current configuration of the analyzed continuum. This method is known as the Updated Lagrangian formulation [10]. However, for surgical simulation, we advocate the Total Lagrangian (TL) formulation of the finite element method in which all variables are referred to the original configuration of the system [33]. The decisive advantage of this formulation is that all derivatives with respect to spatial coordinates are calculated with respect to the original configuration and therefore can be pre-computed as shown in the flowchart of the Total Lagrange Explicit Dynamics (TLED) algorithm presented in Fig. 9.3 (a detailed description of the algorithm is given in [33]).

Following the approach typically applied in explicit dynamics finite element analysis, for computational efficiency of our TLED algorithm we used single point integration for all elements of the mesh (improved linear tetrahedrons [34] and under-integrated linear hexahedrons). Therefore, the nodal forces for each element are computed as:

$$ {}^{t}_{0}\mathbf{F}_{\text{int}} = {}^{t}_{0}\mathbf{X} \cdot {}^{t}_{0}\mathbf{S} \cdot \mathbf{B}_0 \cdot V_0, \tag{9.8} $$

where, according to the notation used in [10], the left superscript represents the current time, the left subscript represents the time of the reference configuration, $\mathbf{F}_{\text{int}}$ is the matrix of nodal forces, $\mathbf{B}_0$ is the matrix of shape function derivatives, $\mathbf{S}$ is the second-Piola Kirchoff stress matrix, $\mathbf{X}$ is the deformation gradient and V_0 is the initial volume. In (9.8), the matrix of shape function derivatives $\mathbf{B}_0$ and the initial volume V_0 are constant and therefore can be pre-computed (see Fig. 9.3).

The main benefits of the TLED algorithm in comparison to the explicit dynamics algorithms using Updated Lagrangian formulation are:

- Allows pre-computing of many variables involved (e.g. derivatives with respect to spatial coordinates), Fig. 9.3
- No accumulation of errors – increased stability for quasi-static solutions
- Second-Piola Kirchoff stress and Green strain are used – appropriate for handling geometric non-linearities
- Easy implementation of the material law for hyperelastic materials using the deformation gradient

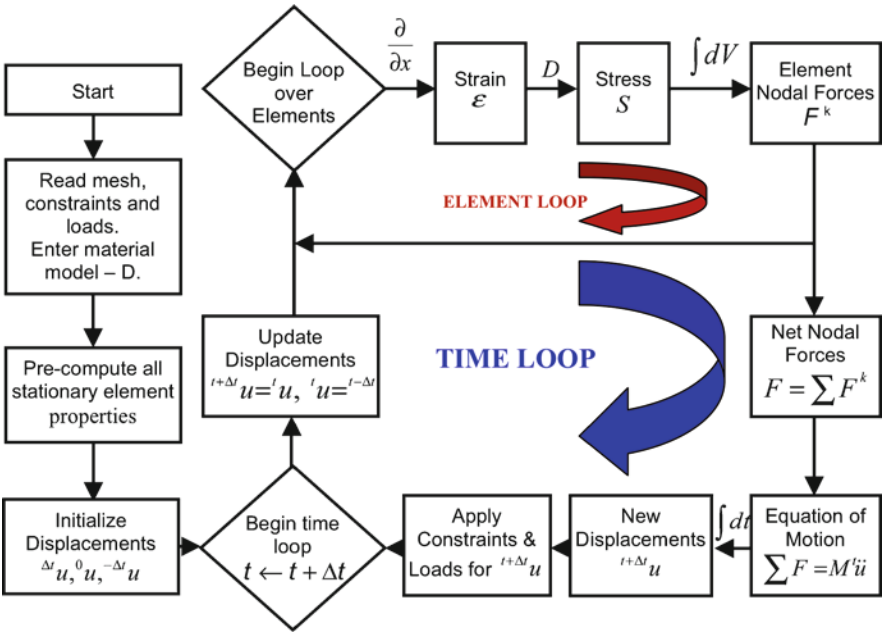

Fig. 9.3 Flowchart of the total lagrange explicit dynamics (TLED) finite element algorithm for surgery simulation. Detailed description of the algorithm is given in [33]

The fact that many quantities involved in the computation of nodal forces can be pre-computed leads to a significant decrease in the computational effort. For instance, the TLED algorithm using eight-noded hexahedral under-integrated elements requires approximately 35% fewer floating-point operations per element, per time step than the Updated Lagrangian explicit algorithm using the same elements [33].

9.4 Algorithms for Neurosurgery Modeling

As explained in Chap. 6, accurate warping of high-quality pre-operative radiographic images to the intra-operative (i.e. deformed) brain configuration in a process known as non-rigid registration is a key element of image-guided neurosurgery. In order to perform such warping, only the final (deformed during surgery) state of the brain needs to be predicted. This requires algorithms for determining steady-state solution for the brain deformations. The deformations at the steady-state must be obtained within the real-time constraints of image-guided neurosurgery, which practically means that the results should be available in 40–80 s.

As stated in Chap. 6, neuroimage registration is a non-linear problem of computational mechanics as it involves large deformations, non-linear material properties and non-linear boundary conditions. However, it is a less demanding problem than surgery simulation as only the steady-state solution for deformations is of interest,

i.e. the time history of forces and deformations does not have to be obtained. Therefore, for image registration, we advocate Dynamic Relaxation (DR), which is an explicit iterative algorithm that relies on the introduction of an artificial mass-dependent damping term in the equation of motion. The damping attenuates the oscillations in the transient response, increasing the convergence towards the steady-state solution. Because of DR's explicit nature, there is no need for solving large systems of equations. All quantities can be treated as vectors, reducing the implementation complexity and the memory requirements. Although the number of iterations to obtain convergence can be quite large, the computation cost of each iteration is very low, making it a very efficient solution method for non-linear problems.

9.4.1 Dynamic Relaxation Algorithm

The basic Dynamic Relaxation (DR) algorithm is presented in [35]. The main idea is to include a mass-proportional damping in the equation of motion (9.1) (see Sect. 9.2), which will increase the convergence speed towards the steady-state. The obtained damped equation is then solved using the central difference method (explicit integration). After the inclusion of mass-proportional damping, the equation of motion (9.1) becomes

$$\mathbf{M}\ddot{\mathbf{u}} + c \cdot \mathbf{M}\dot{\mathbf{u}} + \mathbf{F}(\mathbf{u}) = \mathbf{R}, \tag{9.9}$$

where c is the damping coefficient.

By applying the central difference integration method to the damped equation of motion (9.9), the equation that describes the iterations in terms of displacements becomes:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \beta(\mathbf{u}_n - \mathbf{u}_{n-1}) + \alpha \mathbf{M}^{-1}(\mathbf{R} - \mathbf{F}), \tag{9.10}$$

$$\alpha = 2h^2/(2+ch), \quad \beta = (2-ch)/(2+ch) \tag{9.11}$$

where h is a fixed time increment.

The iterative method defined by (9.10) is explicit as long as the mass matrix is diagonal. As the mass matrix does not influence the deformed state solution, a lumped mass matrix can be used that maximizes the convergence of the method.

In [35], the convergence of the DR algorithm is studied for linear structural mechanics equations, when the nodal forces can be written as

$$\mathbf{F}(\mathbf{u}) = \mathbf{K} \cdot \mathbf{u}, \tag{9.12}$$

with $\mathbf{K}$ being the stiffness matrix.

We extend this study to the non-linear case. We propose to use the linearization of the nodal forces obtained by expanding them in a Taylor series and keeping the first two terms:

$$\mathbf{F}(\mathbf{u}_n) = \mathbf{F}(\mathbf{u}_k) + \mathbf{K}_k \cdot (\mathbf{u}_n - \mathbf{u}_k), \tag{9.13}$$

where $\mathbf{u}_k$ is a point close to $\mathbf{u}_n$ and $\mathbf{K}_k$ is the tangent stiffness matrix evaluated at point $\mathbf{u}_k$.

By substituting $\mathbf{F}$ from (9.13) into (9.10), we obtain the equation that advances to a new iteration for a non-linear problem as:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \beta(\mathbf{u}_n - \mathbf{u}_{n-1}) + \alpha\mathbf{b} - \alpha\mathbf{A}\mathbf{u}_n \tag{9.14}$$

with

$$\mathbf{b} = \mathbf{M}^{-1}(\mathbf{R} - \mathbf{F}(\mathbf{u}_k) + \mathbf{K}_k\mathbf{u}_k), \quad \mathbf{A} = \mathbf{M}^{-1}\mathbf{K}_k. \tag{9.15}$$

It is worth noting that (even if (9.14) has the same form as in the linear case) the point $\mathbf{u}_k$ is not fixed during the iteration process (as it must be close to $\mathbf{u}_n$ in order for the Taylor series expansion to be accurate). Therefore, the tangent stiffness matrix (and matrix $\mathbf{A}$) changes.

The error after the nth iteration is defined as:

$$\mathbf{e}_n = \mathbf{u}_n - \mathbf{u}^*, \tag{9.16}$$

where $\mathbf{u}^*$ is the solution. Substituting (9.16) in (9.14) gives the error equation (valid only close to the solution):

$$\mathbf{e}_{n+1} = \mathbf{e}_n - \alpha\mathbf{A}\mathbf{e}_n + \beta(\mathbf{e}_n - \mathbf{e}_{n-1}) \tag{9.17}$$

and by assuming that

$$\mathbf{e}_{n+1} = \boldsymbol{\kappa} \cdot \mathbf{e}_n \tag{9.18}$$

the following relation is obtained for computing the eigenvalues κ of matrix $\boldsymbol{\kappa}$:

$$\kappa^2 - (1 + \beta - \alpha A)\kappa + \beta = 0, \tag{9.19}$$

where A denotes any eigenvalue of matrix $\mathbf{A}$.

The fastest convergence is obtained for the smallest possible spectral radius $\rho = |\boldsymbol{\kappa}|$. The optimum convergence condition is obtained when:

$$\rho^* = |\kappa^*| = \beta^{1/2} \approx \left|1 - 2\sqrt{\frac{A_0}{A_\mathrm{m}}}\right|, \tag{9.20}$$

$$h \approx 2/\sqrt{A_\mathrm{m}} = 2/\omega_{\max}, \tag{9.21}$$

$$c \approx 2\sqrt{A_0} = 2\omega_0, \tag{9.22}$$

where A_0 and A_m are the minimum and maximum eigenvalues of matrix $\mathbf{A}$ and therefore ω_0 and $\omega_{\max}$ are the lowest and highest circular frequencies of the un-damped equation of motion [35].

The effect of eigenvalue estimation accuracy on the convergence of the method is presented in [36]. To ensure convergence, it is critical that the maximum

eigenvalue A_m is over-estimated, even if this will lead to a decreased convergence speed. If, at the same time, the minimum eigenvalue A_0 is under-estimated, then uniform convergence will be obtained for all eigenvalues, with a further decrease in convergence speed. If A_0 is over-estimated then it is possible to increase the convergence speed for all eigenvalues except a very narrow range of small eigenvalues.

9.4.1.1 Dynamic Relaxation Algorithm: Maximum Eigenvalue A_m and Mass Matrix

Using the Rayleigh quotient, it has been demonstrated that the maximum eigenvalue of an assembled finite element mesh is bounded by the maximum eigenvalue of any of the elements in the mesh [10]:

$$A_m \leq \max(\lambda^e{}_{max}). \tag{9.23}$$

Therefore, an estimation of the maximum eigenvalue can be obtained by estimating the maximum eigenvalue of each element in the mesh. Such estimations for different element types are presented in [37].

In the case of a non-linear problem, the maximum eigenvalue changes during the simulation as the geometry of the elements changes and therefore it must be estimated after every iteration step.

Because the mass matrix has no influence on the steady-state (as the time derivatives in (9.9) become zero), a fictitious mass matrix that improves the convergence rate can be used. The mass matrix can be chosen such that it reduces the condition number of matrix **A**, leading to a decrease in the spectral radius ρ (see (9.20)).

In order to reduce the condition number, we propose to align the maximum eigenvalue of all elements in the mesh to the same value by changing the density of each element. By doing this, we can still use (9.23) for estimating the maximum eigenvalue, and the condition number is at least preserved and generally decreased, as shown in [35]. This process guarantees that the selected maximum eigenvalue A_m is an over-estimation of the actual maximum eigenvalue during the simulation, therefore ensuring the convergence.

9.4.1.2 Dynamic Relaxation Algorithm: Estimation of the Minimum Eigenvalue A_0

Estimating the minimum eigenvalue is difficult, especially for non-linear problems, where an adaptive procedure should be used in order to obtain the optimum convergence parameters. An overview of the procedures proposed by different authors in the context of DR (including an adaptive one) is presented in [35].

The adaptive method proposed in [35] is based on Rayleigh's quotient and the use of a local diagonal stiffness matrix. The elements of this matrix are computed using finite differences, which can be very difficult to do for degrees of freedom which have small displacement variation.

In this section, we propose a new adaptive method for computing the minimum eigenvalue, which is also based on Rayleigh's quotient, but does not have the shortcomings of the method proposed in [35].

We consider a change of variable:

$$\mathbf{z}_n = \mathbf{u}_n - \mathbf{u}_k, \tag{9.24}$$

where $\mathbf{u}_k$ is the point used for linearization of the nodal forces in (9.13). The linearised nodal forces can therefore be expressed as:

$$\mathbf{F}(\mathbf{u}_n) = \mathbf{F}(\mathbf{u}_k) + \mathbf{K}_k \cdot \mathbf{z}_n. \tag{9.25}$$

After replacing (9.24) and (9.25) in (9.1), the linearized equation of motion becomes:

$$\mathbf{M} \cdot \ddot{\mathbf{z}} + \mathbf{K}_k \cdot \mathbf{z} = \mathbf{R} - \mathbf{F}(\mathbf{u}_k). \tag{9.26}$$

We can now rely on (9.26) to estimate A_0 using Rayleigh's quotient and the current value of the displacements:

$$A_0 \leq \frac{(\mathbf{z}_n)^{\mathrm{T}} \mathbf{K}_k \mathbf{z}_n}{(\mathbf{z}_n)^{\mathrm{T}} \mathbf{M} \mathbf{z}_n}. \tag{9.27}$$

We consider the right-hand side of (9.27) as an estimate of the minimum eigenvalue. Using (9.24) and (9.25), this estimate becomes:

$$A_0 \approx \frac{(\mathbf{u}_n - \mathbf{u}_k)^{\mathrm{T}} (\mathbf{F}(\mathbf{u}_n) - \mathbf{F}(\mathbf{u}_k))}{(\mathbf{u}_n - \mathbf{u}_k)^{\mathrm{T}} \mathbf{M} (\mathbf{u}_n - \mathbf{u}_k)}, \tag{9.28}$$

where $\mathbf{u}_k$ is a fixed point that must be close to $\mathbf{u}_n$. We choose the solution from a previous iteration as the fixed point $\mathbf{u}_k$ and update it after a number of steps in order to keep it close to the current solution $\mathbf{u}_n$. No additional information (such as estimates of the stiffness matrix) is required and only vector operations are performed (as $\mathbf{M}$ is a diagonal lumped mass matrix).

During the iterative dynamic relaxation DR procedure, the high frequencies are damped out and the system will eventually oscillate on its lowest frequency. Therefore, (9.28) will converge towards the minimum eigenvalue. This estimation process, combined with our parameter selection process, leads to an increased convergence rate, because it always offers an over-estimation of the minimum eigenvalue. The higher the over-estimation of the minimum eigenvalue, the higher the reduction of the high frequency vibrations (see [36]), and therefore (9.28) will converge faster towards the real minimum eigenvalue.

9.4.1.3 Dynamic Relaxation Algorithm: Termination Criteria

One very important aspect of any finite element (FE) algorithm is the termination criterion used. If the criterion is too coarse, then the solution might be too inaccurate and if the criterion is too tight, then time is lost for unnecessary computations.

The criteria used by commercial FE software are usually based on residual forces, displacements or energy. None of these criteria gives any information about the absolute error in the solution and selecting any of these termination criteria is very difficult.

We propose a new termination criterion that gives information about the absolute error in the solution, particularly suited for our solution method. Because DR iterations lead to a strong reduction of the high frequencies, the displacement vector will oscillate around the solution vector with a frequency that converges towards the smallest oscillation frequency. This implies that the error vector **e** will converge toward the eigenvector corresponding to the lowest eigenvalue. Therefore, we can make the following approximation:

$$\mathbf{A}\cdot\mathbf{e}_n \approx A_0\cdot\mathbf{e}_n. \tag{9.29}$$

By substituting (9.29) in (9.17) and considering relations given in (9.19) and (9.20), we obtain:

$$\mathbf{u}_{n+1}-\mathbf{u}^* \approx \rho\cdot(\mathbf{u}_n-\mathbf{u}^*). \tag{9.30}$$

Therefore, after each iteration step, the error is reduced by a ratio equal to ρ. We can now obtain an approximation of the absolute error in the solution by applying the infinity norm to (9.30):

$$\|\mathbf{u}_{n+1}-\mathbf{u}^*\|_\infty \approx \rho\cdot\|\mathbf{u}_n-\mathbf{u}^*\|_\infty \le \rho\cdot(\|\mathbf{u}_{n+1}-\mathbf{u}^*\|_\infty + \|\mathbf{u}_{n+1}-\mathbf{u}_n\|_\infty), \tag{9.31}$$

$$\|\mathbf{u}_{n+1}-\mathbf{u}^*\|_\infty \le \frac{\rho}{1-\rho}\cdot\|\mathbf{u}_{n+1}-\mathbf{u}_n\|_\infty . \tag{9.32}$$

Therefore, the convergence criterion can be defined as:

$$\frac{\rho}{1-\rho}\cdot\|\mathbf{u}_{n+1}-\mathbf{u}_n\|_\infty \le \varepsilon, \tag{9.33}$$

where ε is the imposed absolute accuracy. This convergence criterion gives an approximation of the absolute error based on the displacement variation norm from the current iteration.

Because our parameter estimation procedures over-estimate the maximum eigenvalue A_m and under-estimate the minimum eigenvalue A_0, the value of the computed spectral radius ρ_c we use in (9.33) can be lower than the real value of the spectral radius (see (9.20)). This can lead to an early termination of the iteration process. Therefore, in (9.33) we use a corrected value of the computed spectral radius:

$$\rho_{co} = \rho_c + \zeta\cdot(1-\rho_c), \tag{9.34}$$

where ζ is a correction parameter with values between 0 and 1, defining the maximum under-estimation error for the spectral radius ρ. In our simulations we use $\zeta=0.2$.

9.5 Element Formulation for Finite Element Algorithms for Surgery Simulation and Neurosurgery Modeling

9.5.1 Volumetric Locking

As stated in Chap. 6, due to stringent computation time requirements, the finite element meshes for models applied in surgery simulation and image registration must be constructed using low-order elements that are computationally inexpensive. Mixed meshes consisting of tetrahedral and hexahedral element are most convenient from the perspective of automation of simulation process. However, the standard formulation of the tetrahedral element exhibits volumetric locking, especially in case of soft tissues such as the brain, which are modeled as almost incompressible materials [38–44]. There are a number of improved linear tetrahedral elements already proposed by different authors [45–48]. The average nodal pressure (ANP) tetrahedral element proposed by Bonet and Burton in [45] is computationally inexpensive and provides much better results for nearly incompressible materials than the standard tetrahedral element. Nevertheless, one problem with the ANP element and its implementation in a finite element code is the handling of interfaces between different materials. In [34], we extended the formulation of the ANP element so that all elements in a mesh are treated in a similar way, requiring no special handling of the interface elements.

9.5.2 Stability of Under-Integrated Hexahedral Elements; Hourglassing

As stated in Chap. 6, low-order hexahedral elements with one Gauss point (referred to in Chap. 1 as a linear under-integrated hexahedral elements) are the preferred choice for explicit dynamics-type algorithms from the perspective of computational efficiency. However, such elements exhibit unphysical zero-energy deformation modes (hourglassing). The hourglass modes can be controlled by calculating hourglass forces that oppose the hourglass deformation modes. We have shown in [49] that the hourglass control forces for each element can be computed (in matrix form) as:

$$ {}^{t}_{0}\mathbf{F}^{\mathrm{Hg}} = k \cdot {}_{0}\mathbf{Y} \cdot {}_{0}\mathbf{Y}^{\mathrm{T}} \cdot {}^{t}_{0}\mathbf{u}, \tag{9.35} $$

where k is a constant that depends on the element geometry and material properties, ${}_0\mathbf{Y}$ is the matrix of hourglass shape vectors and $\mathbf{u}$ is the matrix of current displacements. In (9.35), all quantities except $\mathbf{u}$ are constant and can be pre-computed, which makes the hourglass control mechanism very efficient.

9.6 Modeling of the Brain–Skull Interactions for Image-Guided Neurosurgery: Efficient Finite Sliding Contact Algorithm

Modeling of interactions between continua (e.g. soft organs) undergoing deformations is a challenging task. To facilitate such modeling, many sophisticated contact algorithms have been proposed in the literature (e.g. [50–53]) and implemented in commercial finite element codes such e.g. ABAQUS [26] and LS-DYNA [3]. Application of such algorithms tends to consume significant computing resources, which substantially increases the solution time.

When computing the brain deformation for neuroimage registration, we are interested in the interactions between the brain and the rigid skull that provide constraints for the brain tissue deformation and brain rigid body motion. Accurate modeling of such interactions can be done using a very efficient algorithm that treats these interactions as a finite sliding, frictionless contact between a deformable object (the brain) and a rigid surface (the skull) [54]. The main parts of such a contact algorithm (for detailed description see [54]) are, first, the detection of nodes on the brain surface (also called the slave surface) which have penetrated the skull surface (master surface) and second, the displacement of each slave node that has penetrated the master surface to the closest point on the master surface.

An efficient penetration detection algorithm can be formulated based on the closest master node (nearest neighbor) approach [3]. As the surfaces of the anatomical structures of the segmented neuroimages are typically discretized using triangles, the skull surface can be treated as a triangular mesh. We refer to each triangular surface as a "face", to the vertices – "nodes" and to the triangles' sides – "edges". Using this terminology, the basic brain–skull contact algorithm is described as follows:

– For each slave node P:

  - Find the closest master node C (global search).
  - Check the faces and edges surrounding C for penetration (local search).
  - Check additional faces and edges that might be penetrated by P (identified in the master surface analysis stage – because the master surface is rigid, this analysis can be done pre-operatively).

Further improvement of efficiency of the penetration detection algorithm and computation speed is done by implementing bucket sort [3, 53] in the global search phase.

9.7 Alternatives to Finite Element Method for Image-Guided Neurosurgery and Surgery Simulation: Meshless Algorithms

There are two important factors limiting the application of finite element methods for predicting brain responses in image-guided neurosurgery and surgery simulation:

1. Time-consuming generation of patient-specific finite element meshes of the brain and other body organs (more detailed discussion on generation of patient-specific computational grids is provided in Chap. 6).
2. Deterioration of the solution accuracy and instability as the elements undergo distortion (inversion) when surgical tools induce large deformations [31].

Meshless algorithms [55], in which the analyzed continuum is discretized by nodes (where forces and displacements are calculated) with no assumed structure on the interconnection of the nodes and integration points (where stresses and strains are calculated) (Fig. 9.4), have been proposed in the literature for generating computational grids of domains of complex geometry and providing reliable results for large deformations [56–58].

Smoothed particle hydrodynamics SPH is regarded as the first meshless method. It utilizes a strong form of equations of continuum mechanics, and the nodes are also the integration points [59]. SPH and other particle methods (such as material point method in which a weak form of equations of continuum mechanics is used) were applied in injury biomechanics [60, 61]. However, the literature indicates several important shortcomings of the SPH method. These include instabilities in tension and accuracy inferior to that of the finite element method [55]. Therefore, we

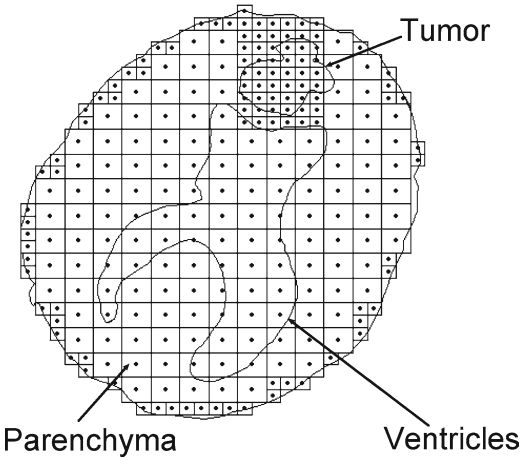

Fig. 9.4 Background regular integration grid for a patient-specific meshless model of the brain with tumor. The integration points are indicated as *black dot*. Note that the background grid does not conform to the geometry boundary. Adapted from [69]

focus on meshless methods that utilize the weak form of the equations of continuum mechanics and background integration grid. As an example, we discuss the Meshless Total Lagrangian Explicit Dynamics algorithm [56] which is motivated by the need for simple, automatic model creation for real-time simulation.

9.7.1 Meshless Total Lagrangian Explicit Dynamics (MTLED): Algorithm Description

In the Meshless Total Lagrangian Explicit Dynamics (MTLED) algorithm, the integration of equations of continuum mechanics is done over a background grid of hexahedral cells with a single integration point per cell (the idea similar to the one used in under-integrated hexahedral elements described in Sect. 9.2) [56] (Fig. 9.4). As this grid does not have to conform to the boundary of the analyzed continuum, it can be generated automatically even for complicated geometry. The nodes where the displacements are calculated are independent of the background integration grid. Almost arbitrary placement of the nodes throughout the analyzed continuum can be used, which is well suited for complicated geometry. However, restrictions (discussed later) on the ratio of the number of integration points and nodes apply.

Construction of the shape functions is the crucial difference between the MTLED algorithm and the Total Lagrangian Explicit Dynamics (TLED) finite element algorithm described in Sect. 9.3. In the MTLED algorithm, we use Moving Least-Squares shape functions that were initially applied in the Diffuse Element Method by Nayroles et al. [62]:

$$\mathbf{u}^h(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}(\mathbf{x}) \cdot \mathbf{a}(\mathbf{x}), \tag{9.36}$$

where u^h is the approximation of the displacement u, $\mathbf{p}(\mathbf{x})$ is the vector of monomial basis function, $\mathbf{a}(\mathbf{x})$ is the vector of coefficients that need to be calculated, and $\boldsymbol{x}$ is the point belonging to the analyzed continuum but not located at the node. In the MTLED algorithm, low-order (up to quadratic order) monomial basis functions are used [56]:

$$\mathbf{p}^{\mathrm{T}}(\mathbf{x}) = (1 \,|\, x\, y\, z \,|\, xy\, xz\, yz \,|\, x^2 y^2 z^2). \tag{9.37}$$

It has been shown that for a stable solution, the nodal support domains need to overlap (i.e. a given node must belong to more than one support domain) and that the number of nodes in a given support domain should be approximately twice the length of the vector of basis functions [56] (for (9.37) the possible vector lengths are 4 for linear basis functions and 10 for quadratic ones). Therefore, from the perspective of solution accuracy, stability and computation efficiency, a trade-off exists between the number of nodes and the total number of integration points. Through parametric study, it has been estimated in [56] that the number of integration points should be twice the number of nodes to ensure accuracy and convergence. According to [56], increasing the number of integration points beyond this ratio exerts negligible effects on the results.

9.8 Real-Time Computations without Supercomputers: Increasing Computation Speed Through Algorithm Implementation on Graphics Processing Unit (GPU)

The algorithms for surgery simulation and image-guided surgery discussed in this chapter facilitate efficient and robust computations. For instance, they make it possible to compute the deformation field within the brain for an image registration problem in under 40 s on a standard personal computer for non-linear finite element models consisting of around 30,000 elements. However, predicting the time history of the force between the soft tissue and the surgical tool at a frequency of 500 Hz (required for haptic feedback) poses a challenge even for very efficient non-linear algorithms deployed on a personal computer.

For hardware-based increase of computation speed, we advocate the implementation of our algorithms on graphics processing units (GPUs). This is done by using a GPU as a coprocessor for the computer's central processing unit (CPU) for executing sections of the code that can be run in parallel. Before the introduction of NVIDIA's Compute Unified Device Architecture (CUDA), general-purpose computations on GPUs were done by recasting the computations in graphic terms and using the graphics pipeline [63]. Therefore, a scientific or general-purpose computation often required a concerted effort by experts in both computer graphics and in the particular scientific or engineering domain. With the introduction of CUDA, in November 2006, NVIDIA proposed a new parallel programming model and instruction set for their GPUs that can be used for performing general-purpose computations. CUDA comes with a software environment that allows developers to use C as a high-level programming language. A minimum set of keywords are used to extend the C language in order to: identify the code that must be run on the GPU as parallel threads, identify each thread (and the block of threads it belongs to) and to organize and transfer the data in the different GPU memory spaces. CUDA also exposes the internal architecture of the GPU and allows direct access to its internal resources. The programmer has more control over the internal hardware resources of the GPU, but this comes at the expense of an increased programming effort compared to a CPU implementation.

The GPU has a highly parallel, multithreaded, many core processor architecture and its cost (under US$3,000 for a general-purpose GPU) is orders of magnitude smaller than that of a supercomputer with a comparable number of parallel threads. GPU architecture is well suited for problems that can be expressed as data-parallel computations with high arithmetic intensity, where the same program is executed on many data elements in parallel. CUDA is a general purpose parallel computing architecture that allows the development of application software that transparently scales with the number of processor cores in the GPU.

Because it only uses vectors, an explicit integration algorithm is perfectly suited for parallel implementation on a GPU. We implemented the Dynamic Relaxation algorithm presented in Sect. 9.4 – on GPU using CUDA. We transferred all the computationally intensive parts of the algorithm (element force computation, displacement vector computation, contact handling, parallel reduction – including infinity norm computation and scalar product of vectors) to the GPU, to take advantage

of its massive parallelism. The code was run on a NVIDIA Tesla C870 computing board, which has 16 multiprocessors with eight scalar processor cores each and single-precision floating-point operations. A detailed description of the implementation can be found in [64]. The GPU implementation performs 2,000 iterations of the brain shift simulation in 1.8 s, offering real-time computation capabilities.

9.9 Algorithm Verification

The general guidelines for verification in computational solid mechanics have been proposed by the American Society of Mechanical Engineers ASME in [65]. These guidelines underscore the importance of establishing confidence through collection of evidence that the solution algorithms are working correctly. As for non-linear problems of computational solid mechanics, analytical solutions typically do not exist. Therefore, we advocate collecting such evidence by comparing the results obtained through new algorithms with the solutions from established algorithms (such as those implemented in commercial finite element codes).

In the following sections, we will present verification results for some of the algorithms described in this chapter: hourglass control, volumetric locking, dynamic relaxation and brain–skull interaction (contact).

9.9.1 Hourglass Control

This verification experiment was artificially designed to compound difficulties associated with hourglass control: large deformations, bending and rigid body motions. A column having a height of 1 m and a square section with the side size 0.1 m was meshed using hexahedral elements (Fig. 9.5a). The mesh has 496 nodes and 270 elements. A Neo-Hookean almost incompressible material model was used, having the mechanical properties similar to those of the brain (mass density of 1,000 kg/m^3, Young's modulus in undeformed state equal to 3,000 Pa and Poisson's ratio 0.49).

The deformation was imposed by constraining the lower face and displacing the upper face of the column, with maximum displacements of 0.5 m in the x direction and 0.3 m in the z direction.

The deformed shape obtained using the TLED algorithm is presented in Fig. 9.5b for the under-integrated hexahedral elements with no hourglass control. The influence of the presented hourglass control mechanism can be clearly seen in Fig. 9.5c.

The displacements of a line of nodes from the side of the column (in the plane $y=0$) are presented in Fig. 9.6. These displacements are compared with the results obtained using the commercial finite element software ABAQUS (fully integrated linear hexahedral elements with hybrid displacement-pressure formulation).

The displacement maximum relative error, defined as the ratio between the maximum displacement difference and the imposed displacement, was 1.4% in the case of column deformation. This demonstrates the good accuracy of the elements using the proposed hourglass control mechanism.

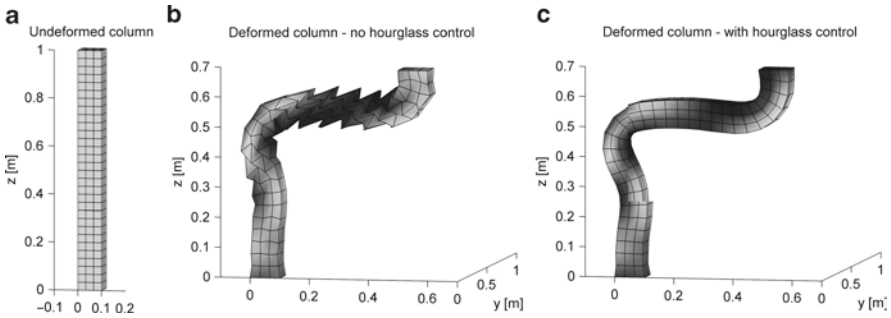

Fig. 9.5 Verification of hourglass control algorithm using deformation of a column as an example. (**a**) Undeformed shape; (**b**) deformed shape with no hourglass control; and (**c**) deformed shape with successful hourglass control. Copied from [49]

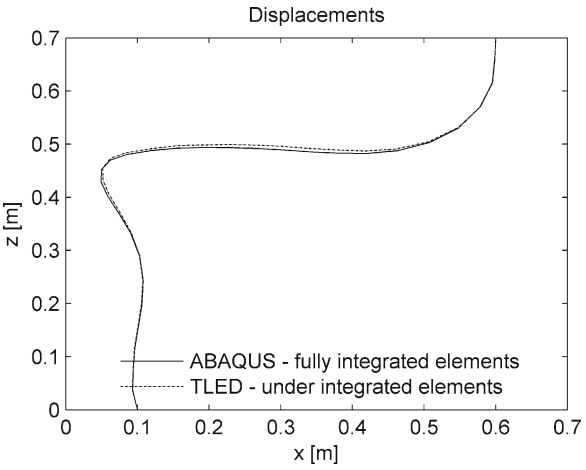

Fig. 9.6 Deformation of a column (*middle line displacements*). Comparison of results between TLED and ABAQUS finite element solver. Copied from [49]

9.9.2 Volumetric Locking

Because the only difference between our improved ANP (IANP) element and the standard ANP element consists in the way interfaces between different materials are handled, we designed a simulation experiment that highlights these differences. We considered a cylinder with a diameter of 0.1 m and a height of 0.2 m made out of alternating sections with two different material properties, as shown in Table 9.1. We used a Neo-Hookean material model for both materials.

Half of the nodes on the upper face of the cylinder were displaced in order to create a complex deformation field at different material interfaces (Fig. 9.7a).

Using the cylindrical geometry, we created a hexahedral mesh (13,161 nodes and 12,000 elements) and a tetrahedral mesh (11,153 nodes and 60,030 elements). The behavior of the following elements was compared:

Table 9.1 Material properties

Property	Material 1	Material 2
Young's modulus E [Pa]	3,000	30,000
Poisson ratio ν	0.49	0.48
Density ρ (kg/m^3)	1,000	1,000

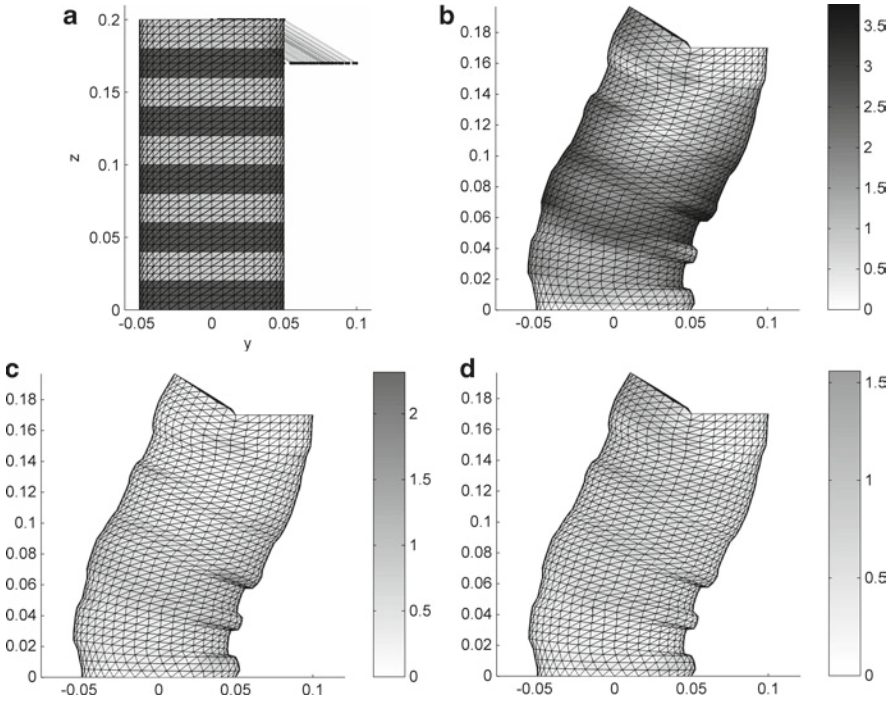

Fig. 9.7 Deformation of a cylinder made out of sections with different material properties. (**a**) The undeformed configuration and the nodal displacements applied. The *color bars* show the difference in positions of the surface nodes, in millimeters, between the models using hexahedral elements and models using (**b**) locking tetrahedral elements (**c**) ANP elements and (**d**) IANP elements. Copied from [34]

1. Fully integrated linear hexahedra, with selectively reduced integration of the volumetric term (Hexa), which should offer a benchmark solution [66]
2. Standard average nodal pressure elements (ANP)
3. Our improved average nodal pressure elements (IANP)
4. Linear standard tetrahedron (Tetra)

All the computations were done using the TLED algorithm. Based on the displacement differences presented in Fig. 9.7, we note that the usage of standard locking tetrahedral elements can lead to errors of up to 3.8 mm in the deformation field. The use of ANP elements reduces the maximum error to 2.3 mm, while the use of IANP elements leads to a maximum error of 1.5 mm (all errors are considered relative to the results of the model that uses Hexa elements).

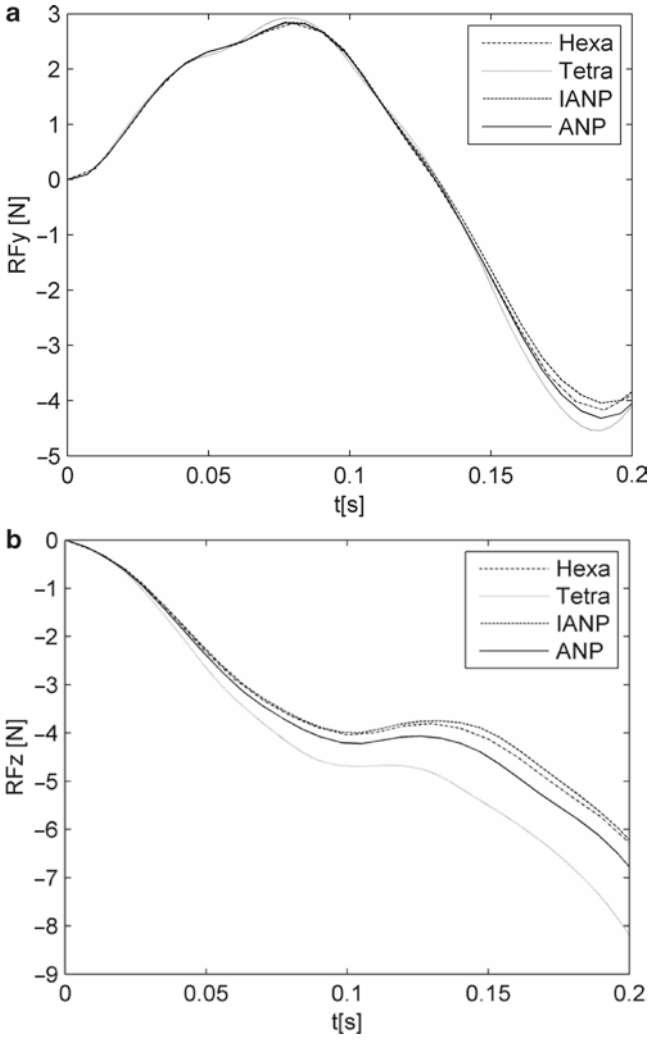

Fig. 9.8 Reaction forces on the displaced face (**a**) in the *y* direction (**b**) in the *z* direction. Adapted from [34]

The reaction forces computed on the displaced face are presented in Fig. 9.8. The results obtained using the IANP elements are the closest to the benchmark results given by the Hexa elements. Therefore, the IANP elements offer the best performances both in terms of displacements and reaction forces, while the standard tetrahedral element offers the worst performances, as expected.

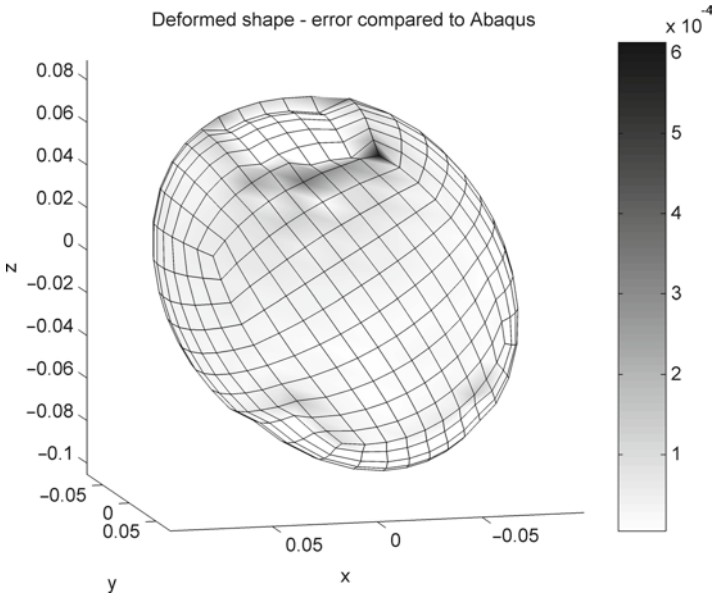

Fig. 9.9 Absolute difference (*gray-scale coded*) in nodal positions between our algorithm and ABAQUS. Dimensions are in meters. Copied from [36]

9.9.3 Dynamic Relaxation: Steady-State Computation

We use this verification example to demonstrate the accuracy of our steady-state computation method. For an ellipsoid having approximately the size of the brain, we fixed a set of nodes (at the bottom) and displaced another set of nodes (at the top) in order to obtain a deformation field similar to what happens in brain shift. The mesh was created using hexahedral elements and has 2,200 elements and 2,535 nodes. We used an almost incompressible Neo-Hookean material model and a large displacement value (2 cm).

We performed the displacement computation first by using our algorithm and second by using ABAQUS [26]. For computational efficiency, we use under-integrated hexahedral elements with the hourglass control implementation based on the relations presented in [49]. In ABAQUS, we used hybrid displacement-pressure hexahedral elements, which are the "gold standard" for almost incompressible materials. We used the static solver with the default configuration and assumed that the ABAQUS simulation provides accurate results.

The error distribution (absolute difference in nodal position between the two simulations) is presented in Fig. 9.9. The maximum error magnitude of 0.6 mm is obtained at the edge of the displaced area and it is mainly an artifact of using under-integrated elements. Nevertheless, the average error is 0.025 mm which demonstrates that our simulation results are more than acceptable (as the error is much smaller than the accuracy of image-guided neurosurgery).

9.9.4 Brain–Skull Interface: Contact Algorithm

In order to assess the performance of our brain–skull interface algorithm, we performed simulations using our implementation of the contact algorithm (combined with Dynamic Relaxation as a solution method) and the commercial finite element solver LS-DYNA [3], and compared the results. The same loading conditions and material models were used for both solvers. The loading consisted of displacements applied to the nodes in the craniotomy area using a smooth loading curve. Neo-Hookean material models were used for the brain and tumor tissues, and a linear elastic model was used for the ventricles. In order to obtain the steady-state solution, the oscillations were damped using both mass and stiffness proportional damping in LS-DYNA.

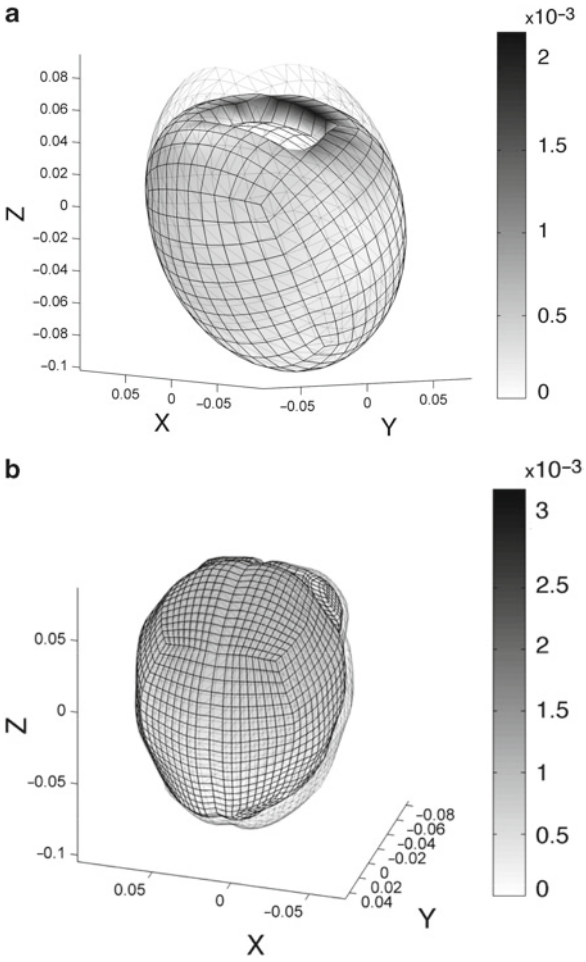

Fig. 9.10 Displacement differences (in millimeters) between our results and LS-DYNA simulations are presented using color codes. The transparent mesh is the master contact. (**a**) Ellipsoid model; (**b**) brain model. Adapted from [54]

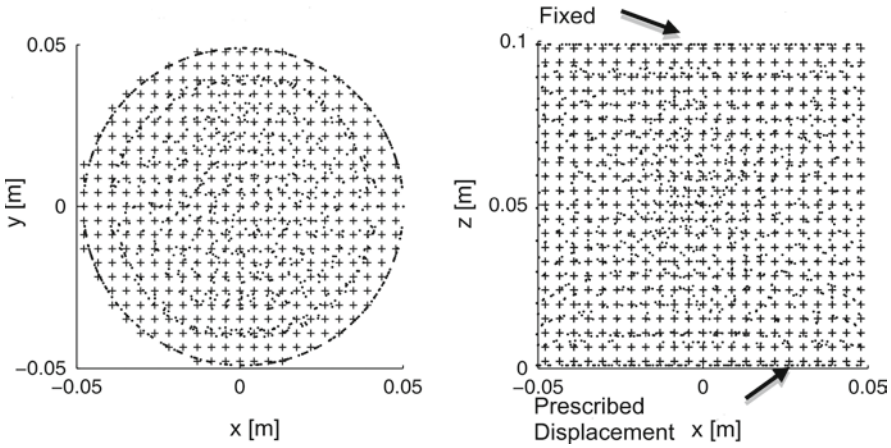

Fig. 9.11 Meshless model of a cylinder used in verification of the MTLED algorithm by Horton et al. [56]. The nodes are indicated as *dot* and integration points as *plus*. Note almost arbitrary node placement. The integration points do not conform to geometry. The boundary conditions are shown in the *right-hand side* figure: the nodes on the *top boundary* were constrained and the prescribed displacement was applied to the nodes on the *bottom boundary*. Adapted from [56]

In a first simulation experiment, we displaced an ellipsoid (made of a hyperelastic Neo-Hookean material) with the approximate size of a brain inside another ellipsoid simulating the skull. The maximum displacement applied was 40 mm. The average difference in the nodal displacement field between our simulation and the LS-DYNA simulation was less than 0.12 mm (Fig. 9.10a).

In the second simulation experiment, we performed the registration of a patient-specific brain shift. The LS-DYNA simulations for this case have been done previously and the results were found to agree well with the real deformations [67]. We performed the same simulations using Dynamic Relaxation and our contact algorithm. The average difference in the nodal displacement field was less than 0.2 mm (Fig. 9.10b).

9.9.5 Meshless Total Lagrangian Explicit Dynamics (MTLED) Algorithm

The MTLED algorithm has been verified by comparing the results obtained using this algorithm with those of an established finite element code (ABAQUS implicit nonlinear solver was used) when modeling semi-confined uniaxial compression and shear of a cylinder made from a very soft (shear modulus of 1 kPa) hyperelastic (Neo-Hookean) material. In meshless discretization of the cylinder, almost arbitrary node placement and integration points non-conforming to geometry were used (Fig. 9.11).

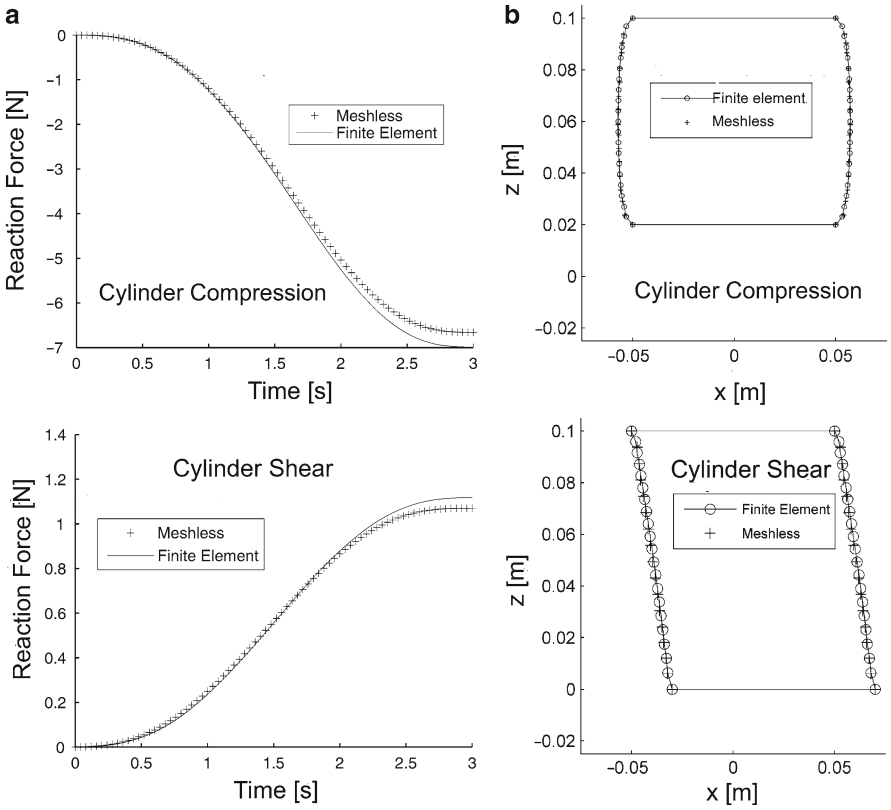

Fig. 9.12 Comparison of the results obtained when modeling 20% compression and shear of a cylinder using meshless (MTLED) and finite element (ABAQUS implicit solver) algorithms. (**a**) Reaction force vs. time; (**b**) contour of the deformed cylinder at time of 3 s. The displacement u was enforced over a period $T=3$ s using a *3-4-5* polynomial that ensures zero velocity and acceleration at time $t=0$ and time $t=T$ [70]. The displacement magnitude was 0.02 in z direction for compression and 0.02 in x direction for shear. x and z directions are defined in Fig. 9.11. Adapted from [56]

For 20% compression and shear of the cylinder, the difference in the total reaction force on the displaced cylinder surface between MTLED and ABAQUS implicit finite element solver was no more than 5% (Fig. 9.12a). The forces obtained using the meshless algorithm were qualitatively similar to those of the finite element method. The maximum relative difference in displacement between MTLED and ABAQUS was around 3.5% (it can be seen in Fig. 9.12b that some of the meshless nodes do not sit exactly on the deformed finite element boundary).

The MTLED algorithm produces stable results even for very large deformations as indicated by the energy – time histories obtained when modeling the cylinder compressed to 20% of its original height (Fig. 9.13). For such large compression, no verification against the ABAQUS finite solver could be done as the finite element solution became unstable.

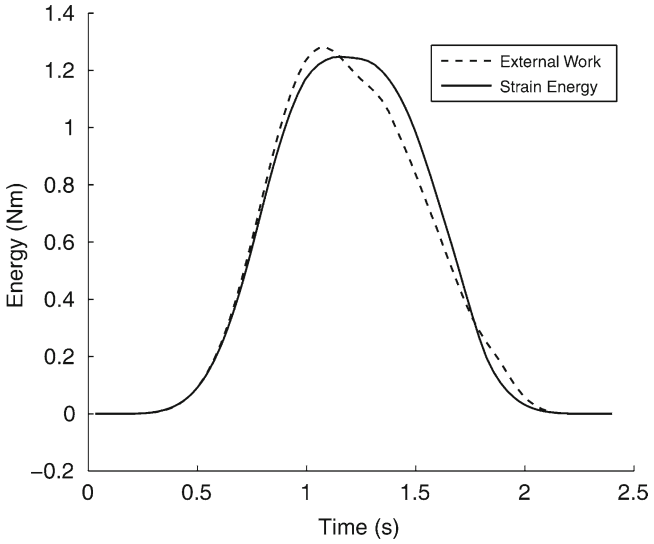

Fig. 9.13 MTLED algorithm. External work and strain energy when compressing a cylinder to 20% of its original height (and returning to the initial state). The displacement was enforced using a *3-4-5* polynomial [70]. Adapted from [56]

9.10 Conclusions

Modeling of the brain for injury simulation and computer-assisted neurosurgery is a non-linear problem of continuum mechanics and involves large deformations, very large strains, non-linear material models, complex loading and boundary conditions and complex geometry. Various finite element (FE) algorithms have been applied for solving this problem.

Modeling of the brain for injury simulation has been often conducted with the idea in mind that numerical surrogates of the human head can be used in the design of countermeasures for traumatic brain injury mitigation. Such modeling has been almost exclusively conducted using non-linear explicit dynamics (i.e. utilizing explicit integration in time domain) finite element algorithms implemented in commercial finite element codes that are routinely used in the automotive industry for transient dynamics problems involving rapid (impact-type) loading such as car structure responses during collision and metals sheet forming.

However, the computational efficiency of the algorithms available in commercial finite element codes is insufficient for computer-integrated neurosurgery where the solution needs to be provided within the real-time constraints of neurosurgery. This led to development of specialized non-linear finite element algorithms aiming at satisfying these constraints. We advocate the application of non-linear finite element algorithms utilizing explicit integration in the time domain (and therefore requiring no iteration for non-linear problems) and Total Lagrangian incremental

formulation of continuum mechanics (as it allows pre-computing of the derivatives with respect to the spatial coordinates):

- Total Lagrangian explicit dynamics (TLED) finite element algorithm for time accurate solution for surgery simulation.
- Dynamic Relaxation (DR) Total Lagrangian algorithm for computing steady-state deformations for neurosurgical modeling.

For hardware-based increase of computation speed, we propose the implementation of these algorithms on graphics processing units (GPUs) by using a GPU as a coprocessor for the computer's CPU. It has been shown in [64] that the implementation of the finite element Dynamic Relaxation algorithm on NVIDIA Tesla C870 GPU performs 2,000 iterations of the brain shift simulation in under 2 s, offering real-time computation capabilities at a fraction of a traditional supercomputer or PC cluster cost. It can be expected that for newer generation of GPUs this already excellent performance would appreciably improve due to significant increase in the number of streaming processor cores (e.g. NVIDIA Tesla C870 GPU had 128 cores while NVIDIA Fermi GPUs released in 2009 have 512 cores [68]) and available shared memory (e.g. 16 kb for NVIDIA Tesla and 48 kb for NVIDIA Fermi).

Application of the most efficient finite element algorithms in surgery simulation and neurosurgery modeling is limited by time-consuming generation of patient-specific finite element meshes and deterioration of the solution accuracy when the elements undergo distortion induced by large deformations. As a solution for overcoming these limitations, we advocate meshless algorithms in which the computational grid has the form of a "cloud" of points. The Meshless Total Lagrangian Explicit Dynamics (MTLED) algorithm described in this chapter has been proven to be capable of providing a stable solution in situations (such as compressing of a cylinder made of soft hyperelastic material to 20% of its original height) where the well-established and extensively verified non-linear finite element algorithms fail.

The instabilities due to element distortion are not limited to simulation and neurosurgery modeling using finite element algorithms. They are even more pronounced in modeling for injury simulation where the brain is exposed to transient loads due to rapid acceleration and/or direct impact to the head. Therefore, although the Meshless Total Lagrangian Explicit Dynamics algorithm described in this chapter was developed in the context of real-time patient-specific surgery simulation, its ability to provide a stable solution for very large strains is relevant also for injury simulation. Development of meshless algorithms that facilitate modeling of surgical dissection and injury-related rupture of soft tissues provides the next challenge in injury and surgery simulation.

Acknowledgements The financial support of the Australian Research Council (grants no. DP0343112, DP0664534, DP1092893 and LX0560460) and NIH (grant no. R03-CA126466-01A1) is greatly acknowledged.

We thank our collaborators Dr Ron Kikinis and Dr Simon K. Warfield of Harvard Medical School (Boston, MA, USA), and Dr Kiyoyuki Chinzei and Dr Toshikatsu Washio of Surgical Assist Technology Group of AIST (Tsukuba, Japan) for help in various aspects of this work.

References

1. Fressmann, D., Munz, T., Graf, O., et al.: FE human modelling in crash – aspects of the numerical modelling and current applications in the automotive industry. LS-DYNA Anwenderforum. DYNAmore GmbH, Frankthenhal, Germany, pp. F-I-23–F-I-34 (2007)
2. Takhounts, E.G., Eppinger, R.H., Campbell, J.Q., et al.: On the development of the SIMon finite element head model. Stapp Car Crash J. **47**, 107–133 (2003)
3. Hallquist, J.O.: LS-DYNA Theory Manual. Livermore Software Technology Corporation, Livermore (2005)
4. PSI: Pam-System Programs in Applied Mechanics. PAM-CRASH, PAM-SAFE Version 1998. Release Note. ESI Group Software Product Company (1998)
5. Mecalog: RADIOSS Users' Manual, Paris, France (1994)
6. Baumann, R., Glauser, D., Tappy, D., et al.: Force feedback for virtual reality based minimally invasive surgery simulator. Stud. Health Technol. Inform. **29**, 564–579 (1996)
7. Cover, S.A., Ezquerra, N.F., O'Brien, J.F., et al.: Interactively deformable models for surgery simulation. IEEE Comput. Graph. Appl. **13**, 68–75 (1993)
8. Kühnapfel, U., Çakmak, H.K., Maaß, H.: Endoscopic surgery training using virtual reality and deformable tissue simulation. Comput. Graph. **24**, 671–682 (2000)
9. Cotin, S., Delingette, H., Ayache, N.: A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. Vis. Comput. **16**, 437–452 (2000)
10. Bathe, K.-J.: Finite Element Procedures. Prentice-Hall, New Jersey (1996)
11. Bro-Nielsen, M.: Finite element modeling in surgery simulation. Proc. IEEE **86**, 490–503 (1998)
12. Bro-Nielsen, M., Cotin, S.: Real-time volumetric deformable models for surgery simulation using finite elements and condensation. Comput. Graphics Forum **15**, 57–66 (1996)
13. DiMaio, S.P., Salcudean, S.E.: Interactive simulation of needle insertion models. IEEE Trans. Biomed. Eng. **52**, 1167–1179 (2005)
14. Warfield, S.K., Talos, F., Tei, A., et al.: Real-time registration of volumetric brain MRI by biomechanical simulation of deformation during image guided neurosurgery. Comput. Vis. Sci. **5**, 3–11 (2002)
15. Roberts, D.W., Hartov, A., Kennedy, F.E., et al.: Intraoperative brain shift and deformation: a quantitative analysis of cortical displacement in 28 cases. Neurosurgery **43**, 749–758 (1998)
16. Miller, K., Wittek, A.: Neuroimage registration as displacement – zero traction problem of solid mechanics (lead lecture). In: Miller, K., Poulikakos, D. (eds.) Proc. of Computational Biomechanics for Medicine I. Workshop affiliated with Medical Image Computing and Computer-Assisted Intervention MICCAI 2006, Copenhagen, Denmark, 1st October p. 2–13, Samfundslitteratur Grafik, Copenhagen, ISBN 10: 87-7611-149-0 (2006)
17. Wittek, A., Hawkins, T., Miller, K.: On the unimportance of constitutive models in computing brain deformation for image-guided surgery. Biomech. Model. Mechanobiol. **8**, 77–84 (2009)
18. Belytschko, T.: A survey of numerical methods and computer programs for dynamic structural analysis. Nucl. Eng. Des. **37**, 23–34 (1976)
19. Bathe, K.-J.: Crush simulation of cars with FEA. Mechanical engineering http://www.memagazine.org/backissues/november98/features/crushcar.html (1998)
20. Cook, R.D., Malkus, D.S., Plesha, M.E.: Finite elements in dynamics and vibrations. In: Concepts and Applications of Finite Element Analysis, pp. 367–428. Wiley, New York (1989)
21. Hallquist, J.O.: LS-DYNA Theoretical Manual. Livermore Software Technology Corporation, Livemore (1998)
22. Belytschko, T.: An overview of semidiscretization and time integration procedures. In: Belytschko, T., Hughes, T.J.R. (eds.) Computational Methods for Transient Analysis, vol. 1, pp. 1–66. North-Holland, Amsterdam (1983)
23. Dassault Systèmes Simulia Corp.: ABAQUS Theory Manual Version 6.5, Providence, RI, USA (2010)
24. Olovsson, L., Simonsson, K., Unosson, M.: Selective mass scaling for explicit finite element analyses. Int. J. Numer. Methods Eng. **63**, 1436–1445 (2005)

25. Majumder, S., Roychowdhury, A., Subrata, P.: Three-dimensional finite element simulation of pelvic fracture during side impact with pelvis-femur-soft tissue complex. Int. J. Crashworthiness **13**, 313–329 (2008)
26. Hibbitt, D., Karlsson, B., Sorensen, P.: Abaqus Analysis User's Manual Version 6.5. ABAQUS Inc., Providence, RI (2005)
27. Flanagan, D.P., Belytschko, T.: A uniform strain hexahedron and quadrilateral with orthogonal hourglass control. Int. J. Numer. Methods Eng. **17**, 679–706 (1981)
28. Cifuentes, A.O., Kalbag, A.: A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. Finite Elem. Anal. Des. **12**, 313–318 (1992)
29. Cook, R.D., Malkus, D.S., Plesha, M.E.: Concepts and Applications of Finite Element Analysis. Wiley, New York (1989)
30. Hughes, T.J.R.: The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover, Mineola (2000)
31. Wittek, A., Dutta-Roy, T., Taylor, Z., et al.: Subject-specific non-linear biomechanical model of needle insertion into brain. Comput. Methods Biomech. Biomed. Eng. **11**, 135–146 (2008)
32. Rojek, J., Oñate, E., Postek, E.: Application of explicit FE codes to simulation of sheet and bulk metal forming processes. J. Mater. Process. Technol. **80–81**, 620–627 (1998)
33. Miller, K., Joldes, G., Lance, D., et al.: Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. Commun. Numer. Methods Eng. **23**, 121–134 (2007)
34. Joldes, G.R., Wittek, A., Miller, K.: Non-locking tetrahedral finite element for surgical simulation. Commun. Numer. Methods Eng. **25**, 827–836 (2009)
35. Underwood, P.: Dynamic relaxation. In: Belytschko, T., Hughes, T.J.R. (eds.) Computational Methods for Transient Analysis, vol. 1, pp. 245–265. New-Holland, Amsterdam (1983)
36. Joldes, G.R., Wittek, A., Miller, K.: Computation of intra-operative brain shift using dynamic relaxation. Comput. Meth. Appl. Mech. Eng. **198**, 3313–3320 (2009)
37. Hughes, T.J.R.: Analysis of transient algorithms with particular reference to stability behavior. In: Belytschko, T., Hughes, T.J.R. (eds.) Computational Methods for Transient Analysis, vol. 1, pp. 67–155. New-Holland, Amsterdam (1983)
38. Miller, K.: Biomechanics of the Brain for Computer Integrated Surgery. Publishing House of Warsaw University of Technology, Warsaw (2002)
39. Miller, K., Chinzei, K.: Mechanical properties of brain tissue in tension. J. Biomech. **35**, 483–490 (2002)
40. Miller, K., Chinzei, K., Orssengo, G., et al.: Mechanical properties of brain tissue in-vivo: experiment and computer simulation. J. Biomech. **33**, 1369–1376 (2000)
41. Miller, K.: Constitutive modelling of abdominal organs. J. Biomech. **33**, 367–373 (2000)
42. Miller, K., Chinzei, K.: Constitutive modelling of brain tissue; experiment and theory. J. Biomech. **30**, 1115–1121 (1997)
43. Bilston, L.E., Liu, Z., Phan-Tien, N.: Linear viscoelastic properties of bovine brain tissue in shear. Biorheology **34**, 377–385 (1997)
44. Margulies, S.S., Thibault, L.E., Gennarelli, T.A.: Physical model simulations of brain injury in the primate. J. Biomech. **23**, 823–836 (1990)
45. Bonet, J., Burton, A.J.: A simple averaged nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. Commun. Numer. Methods Eng. **14**, 437–449 (1998)
46. Bonet, J., Marriott, H., Hassan, O.: An averaged nodal deformation gradient linear tetrahedral element for large strain explicit dynamic applications. Commun. Numer. Methods Eng. **17**, 551–561 (2001)
47. Zienkiewicz, O.C., Rojek, J., Taylor, R.L., et al.: Triangles and tetrahedra in explicit dynamic codes for solids. Int. J. Numer. Methods Eng. **43**, 565–583 (1998)
48. Dohrmann, C.R., Heinstein, M.W., Jung, J., et al.: Node-based uniform strain elements for three-node triangular and four-node tetrahedral meshes. Int. J. Numer. Methods Eng. **47**, 1549–1568 (2000)
49. Joldes, G.R., Wittek, A., Miller, K.: An efficient hourglass control implementation for the uniform strain hexahedron using the total Lagrangian formulation. Commun. Numer. Methods Eng. **24**, 315–323 (2008)

50. Hallquist, J.O., Goudreau, G.L., Benson, D.J.: Sliding interfaces with contact-impact in large-scale Lagrangian computations. Comput. Meth. Appl. Mech. Eng. **51**, 107–137 (1985)
51. Doghri, I., Muller, A., Taylor, R.L.: A general three-dimensional contact procedure for implicit finite element codes. Eng. Comput. **15**, 233–259 (1998)
52. Stewart, J.R., Gullerud, A.S., Heinstein, M.W.: Solution verification for explicit transient dynamics problems in the presence of hourglass and contact forces. Comput. Meth. Appl. Mech. Eng. **195**, 1499–1516 (2006)
53. Sauvé, R.G., Morandin, G.D.: Simulation of contact in finite deformation problems – algorithm and modelling issues. Int. J. Mech. Mater. Des. **1**, 287–316 (2004)
54. Joldes, G., Wittek, A., Miller, K., et al.: Realistic and efficient brain-skull interaction model for brain shift computation. In: Miller, K., Nielsen, P.M.F. (eds.) Proceedings of Computational Biomechanics for Medicine III. Workshop affiliated with International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI 2008, New York, USA, pp. 95–105, September 2008
55. Belytschko, T., Krongauz, Y., Organ, D., et al.: Meshless methods: an overview and recent developments. Comput. Meth. Appl. Mech. Eng. **139**, 3–47 (1996)
56. Horton, A., Wittek, A., Joldes, G.R., et al.: A meshless total Lagrangian explicit dynamics algorithm for surgical simulation. Int. J. Numer. Methods Biomed. Eng. **26**, 977–998 (2010)
57. Doblare, M., Cueto, E., Calvo, B., et al.: On the employ of meshless methods in biomechanics. Comput. Meth. Appl. Mech. Eng. **194**, 801–821 (2005)
58. Horton, A., Wittek, A., Miller, K.: Subject-specific biomechanical simulation of brain indentation using a meshless method. Proc. of International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI 2007; In Lect. Notes Comput. Sci. **4791**, 541–548 (2007)
59. Monaghan, J.J.: Smoothed particle hydrodynamics. Annu. Rev. Astron. Astrophys. **30**, 543–574 (1992)
60. Lee, S.-H., Darvish, K., Lobovsky, L.: Fluid-structure interaction in finite element modeling of traumatic aortic rupture. ASME Conference Proceedings ASME 2004, Advances in Bioengineering, Anaheim, CA, USA, pp. 337–338, 13–19 November 2004
61. Ionescu, I., Guilkey, J., Berzins, M., et al.: Computational simulation of penetrating trauma in biological soft tissues using the material point method. Stud. Health Technol. Inform. **111**, 213–218 (2005)
62. Nayroles, B., Touzot, G., Villon, P.: Generalizing the finite element method: diffuse approximation and diffuse elements. Comput. Mech. **10**, 307–318 (1992)
63. Owens, J.D., Luebke, D., Govindaraju, N., et al.: A survey of general-purpose computation on graphics hardware. Comput. Graphics Forum **26**, 80–113 (2007)
64. Joldes, G.R., Wittek, A., Miller, K.: Real-time nonlinear finite element computations on GPU – Application to neurosurgical simulation. Comput. Meth. Appl. Mech. Eng. **199**, 3305–3314 (2010)
65. ASME PTC 60/V&V 10: Guide for Verification and Validation in Computational Solid Mechanics. The American Society of Mechanical Engineers (ASME) Standards Committee on Verification and Validation in Computational Solid Mechanics PTC 60/V&V 10. http://cstools.asme.org/csconnect/pdf/CommitteeFiles/24816.pdf (2006)
66. ABAQUS: ABAQUS Theory Manual, Version 5.8. Hibbitt, Karlsson & Sorensen, Inc., Rhode Island (1998)
67. Wittek, A., Miller, K., Kikinis, R., Warfield, S.K.: Patient-specific model of brain deformation: application to medical image registration. J. Biomech. **40**, 919–929 (2007)
68. NVIDIA: NVIDIA's Next Generation CUDA Compute Architecture: Fermi. NVIDIA. http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf (2009)
69. Horton, A., Wittek, A., Miller, K.: Computer simulation of brain shift using an element free Galerkin method. In: Middleton, J., Jones, M. (eds.) 7th International Symposium on Computer Methods in Biomechanics and Biomedical Engineering CMBEE 2006, Antibes, France (2006)
70. Waldron, K.J., Kinzel, G.L.: Kinematics, Dynamics, and Design of Machinery. Wiley, New York (1999)