Applied Mathematical Modelling 39 (2015) 3913–3923

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

A Total Lagrangian based method for recovering the un-deformed configuration in finite elasticity

Grand Roman Joldes [*], Adam Wittek, Karol Miller

Intelligent Systems for Medicine Laboratory, School of Mechanical and Chemical Engineering, The University of Western Australia, 35 Stirling Highway, Crawley/Perth, WA 6009, Australia

A R T I C L E   I N F O

Article history:
Received 18 March 2014
Received in revised form 5 November 2014
Accepted 4 December 2014
Available online 18 December 2014

Keywords:
Un-deformed configuration
Total Lagrangian formulation
Explicit time integration
Dynamic relaxation
Newton–Raphson iterations

A B S T R A C T

The problem of finding the un-deformed configuration of an elastic body, when the deformed configuration and the loads are known, occurs in many engineering applications. Standard solution methods for such problems include conservation laws based on Eshelby's energy–momentum tensor and re-parameterization of the standard equilibrium equations. In this paper we present a different method for solving such problems, based on a re-parameterization of the nodal forces using the Total Lagrangian formulation. The obtained nonlinear system of equations describing equilibrium can be solved using either Newton–Raphson or an explicit dynamic relaxation algorithm. The solution method requires only minor modifications to similar algorithms designed for forward motion calculations. Several examples involving large deformations and different boundary conditions and loads are presented.

1. Introduction

The problem of finding the un-deformed shape of an elastic body, when the final deformed shape, material behavior, applied loads and boundary conditions are known, is encountered in many different engineering applications. A classic example is the design of rubber seals, which are pressed into a channel and need to exert a desired pressure onto the channel. Another application is the design of rubber forms to be used in pressing thin sheets of metal in stamping procedures [1].

This problem stated above has been called by some authors an "inverse elastostatics" problem [2–4]. As many elasticity problems related to model parameter identification are also called inverse problems, in order to avoid confusion we will use the term "direct deformation" when referring to the computation of deformation in an elastic body under known loads starting from the un-deformed configuration and the term "inverse deformation" when referring to the computation of deformations in an elastic body under known loads starting from the deformed configuration.

More recently the need for solutions to inverse deformation problems was also identified in various areas of bio-engineering. In [5] an application to breast biomechanics is presented. The imaging of the breast is performed with the patient lying in a prone position (on the stomach). Surgical procedures such as breast biopsy are usually performed with the patients lying on their back (supine position). The reference un-deformed state of the breast is needed in order to predict the breast shape and perform image registration to the new patient position. In [2,3,6,7] the solution of an inverse deformation problem is used in order to assess stresses in the walls of aneurysms. Sometimes the solution to an inverse deformation problem is avoided by

* Corresponding author. Tel.: +61 8 6488 3125; fax: +61 8 6488 1024.
E-mail address: grand.joldes@uwa.edu.au (G.R. Joldes).

http://dx.doi.org/10.1016/j.apm.2014.12.013
0307-904X/Crown Copyright © 2014 Published by Elsevier Inc. All rights reserved.

modeling the structure in such a way that the stresses can be computed directly based on the geometry and applied loads. For example, aneurysms are modeled as membranes in [8].

A solution method for inverse deformation problems in finite elasticity was proposed in [9], by formulating the problem as a set of balance equations written in terms of inverse deformation and standard boundary conditions. This formulation exploits a set of duality relations that allow the formulation of an inverse deformation problem in a form that appears similar to a standard elastostatic problem. Later, Chadwick [10] re-formulated the equilibrium equations in terms of Eshelby's energy–momentum tensor [11].

In [4] Govindjee and Mihalic show that the energy–momentum formulation has several deficiencies: it places strong continuity requirements on the motion, Eshelby's tensor lacks direct physical connection to the stated problem creating difficulties with the boundary conditions and it cannot handle body forces. They propose a new method based on the re-parameterization of the equilibrium equations, which eliminates these difficulties. The method is later extended to incompressible materials in [1], and was shown to be consistent with the approach presented in [12].

The methods presented so far require the writing of the stress equilibrium equations in terms of the deformed configuration (Eulerian description). Conventional forward finite elasticity analyses are typically formulated with respect to a Lagrangian frame of reference (based on the un-deformed configuration) [13]. Rajagopal et al. present a method of solving inverse deformation problems using a Lagrangian frame of reference in [5]. Their method is based on a finite difference approximation of the Jacobian for the system of equations describing the equilibrium, considering the parameters of the deformed state as an initial estimate for the parameters of the reference state. In [7] Riveros et al. propose a method which uses the displacements obtained by solving the direct problem using the current configuration to iteratively update the geometry, which should converge towards the un-deformed geometry.

In this paper we present a new method of solving inverse deformation problems. The starting point for our method consists of the standard equilibrium equations discretised using a Total Lagrangian (TL) framework. The direct elastostatic problem can be stated, after discretisation, as an equation defining the equilibrium of forces with the displacements as unknowns. The TL framework allows a direct relationship between the un-deformed and the deformed configurations to be defined. We exploit this relationship to rewrite the force equilibrium equation based on the known deformed configuration. The obtained non-linear system of equations can be solved using Newton–Raphson techniques, and we present the relations needed for the construction of the exact stiffness matrix required. We also present a different solution method based on explicit time integration [14] and dynamic relaxation [15].

The paper is organized as follows: the derivation of the proposed method is presented in Section 2, several examples involving different boundary conditions and loads are presented in Section 3, followed by discussion and conclusions in Section 4.

2. Problem formulation and solution method

2.1. Derivation of equilibrium equation

We consider hyperelastic materials for which the internal stresses and strains depend only on the deformation field within the material and are independent of the way the deformation was applied (path-independent). For such materials the constitutive behavior is usually defined using a strain energy potential function:

$$ {}_0^tW = f(I_1, I_2, J, \ldots), \tag{1}$$

where I_1 and I_2 are the first and second strain invariants, computed based on the stretch matrix (Cauchy–Green strain tensor) and J is the total volume change, computed as the determinant of the deformation gradient [13]. The left subscript identifies the starting configuration (in this case the un-deformed state, 0) and the left superscript identifies the current configuration. For anisotropic materials there may be other parameters describing the anisotropic behavior (such as the direction of fibers). Our derivation is based on the isotropic case, but can be easily extended to handle anisotropy.

Based on the strain energy potential the second Piola–Kirchhoff stress, $\mathbf{S}$, can be computed as:

$$ {}_0^t\mathbf{S}({}_0^t\mathbf{F}) = \frac{\partial {}_0^tW}{\partial {}_0^t\mathbf{E}}, \tag{2}$$

where $\mathbf{E}$ is the Green–Lagrange strain. Because both the strain energy potential and the Green–Lagrange strain are defined based on the deformation gradient $\mathbf{F}$, the second Piola–Kirchhoff stress can be considered as a function of the deformation gradient. The deformed configuration of an object can be characterized using the equilibrium between the internal forces in the final deformed state f and the externally applied forces:

$$ {}_0^f\mathbf{R_{int}} = {}_f\mathbf{R_{ext}}. \tag{3}$$

After the weak form of the equilibrium equations is discretised (for example using a finite element discretisation), the deformation field can be described using the nodes of the discretisation and their associated shape functions. The internal forces can be expressed, after discretisation using the Total Lagrangian formulation, as [16,17]:

$$ {}_0^f\mathbf{R}_{\mathbf{int}} = \int_{V_0} {}_0\mathbf{D} \cdot {}_0^f\mathbf{S}^{\mathbf{T}} \cdot {}_0^f\mathbf{F}^{\mathbf{T}} dV_0, \tag{4} $$

where $\mathbf{F}$ is the deformation gradient, V_0 is the domain in the un-deformed configuration and $\mathbf{D}$ is the matrix of shape function derivatives in the un-deformed configuration:

$$ {}_0\mathbf{D} = \frac{\partial_0\mathbf{H}}{\partial_0\mathbf{x}}. \tag{5} $$

If only the final configuration and the corresponding external loads are known, the un-deformed configuration (and the displacements) must be computed by solving the inverse deformation problem. We consider a discretization of the object in the final configuration using a set of n points ${}_f x^k$. Using these points we can define an interpolation function to represent the displacements at any point in the object (using a finite element interpolant):

$$ {}_f\mathbf{u}(\mathbf{x}) = \sum_{k=1}^{n} {}_f\mathbf{H}_k(\mathbf{x})_f\mathbf{u}^k = {}_f\mathbf{U}^T{}_f\mathbf{H}. \tag{6} $$

If we consider the transformation that defines the transition from the original to the deformed configuration, $\varphi{:}V_0 \to V_f$, the position of the discretizing nodes in the un-deformed configuration is given by:

$$ {}_0\mathbf{x}^k = \varphi^{-1}({}_f\mathbf{x}^k). \tag{7} $$

Based on these points we define the shape functions in the un-deformed configuration as:

$$ {}_0\mathbf{H}({}_0\mathbf{x}) = {}_f\mathbf{H}({}_f\mathbf{x}). \tag{8} $$

The matrix of shape function derivatives from Eq. (5)can therefore be expressed as:

$$ {}_0\mathbf{D} = \frac{\partial_0\mathbf{H}}{\partial_0\mathbf{x}} = \frac{\partial_f\mathbf{H}}{\partial_f\mathbf{x}}\frac{\partial_f\mathbf{x}}{\partial_0\mathbf{x}} = \frac{\partial_f\mathbf{H}}{\partial_f\mathbf{x}}{}_0^f\mathbf{F} = {}_f\mathbf{D}{}_0^f\mathbf{F}. \tag{9} $$

The positions of the points in the un-deformed configuration of the object can be obtained by subtracting the deformations from the final positions, as:

$$ {}_0\mathbf{x} = {}_f\mathbf{x} - {}_f\mathbf{u}, \tag{10} $$

and, therefore,

$$ {}_0^f\mathbf{F} = \frac{\partial_0{}^{\mathbf{x}}}{\partial_f\mathbf{x}} = \frac{\partial({}_f\mathbf{x} - {}_f\mathbf{u})}{\partial_f\mathbf{x}} = \mathbf{I} - \frac{\partial_f\mathbf{u}}{\partial_f\mathbf{x}} = \mathbf{I} - \frac{\partial({}_f\mathbf{U}^T{}_f\mathbf{H})}{\partial_f\mathbf{x}} = \mathbf{I} - {}_f\mathbf{U}^T\frac{\partial_f\mathbf{H}}{\partial_f\mathbf{x}} = \mathbf{I} - {}_f\mathbf{U}^T{}_f\mathbf{D}, \tag{11} $$

where $\mathbf{I}$ is the identity matrix. Based on the properties of the deformation gradient [13]:

$$ {}_0^f\mathbf{F} = \left({}_f^0\mathbf{F}^{-\mathbf{1}}\right) = \left(\mathbf{I} - {}_f\mathbf{U}^T{}_f\mathbf{D}\right)^{-1}. \tag{12} $$

Finally, by using the change of variable (10) in Eq. (4), the internal forces can be expressed as:

$$ {}_0^f\mathbf{R}_{\mathbf{int}} = \int_{V_f} {}_0\mathbf{D} \cdot {}_0^f\mathbf{S}^{\mathbf{T}} \cdot {}_0^f\mathbf{F}^{\mathbf{T}} \left| det\left(\frac{\partial_0\mathbf{x}}{\partial_f\mathbf{x}}\right)\right| dV_f = \int_{V_f} {}_0\mathbf{D} \cdot {}_0^f\mathbf{S}^{\mathbf{T}} \cdot {}_0^f\mathbf{F}^{\mathbf{T}} J^{-1} dV_f. \tag{13} $$

Considering Eqs. (2) and (9), Eq. (13) can be re-written as:

$$ {}_0^f\mathbf{R}_{\mathbf{int}} = \int_{V_f} {}_f\mathbf{D} \cdot {}_0^f\mathbf{F} \cdot {}_0^f\mathbf{S}^{\mathbf{T}} \cdot ({}_0^f\mathbf{F}) \cdot {}_0^f\mathbf{F}^{\mathbf{T}} J^{-1} dV_f, \tag{14} $$

with the deformation gradient computed based on the final configuration using Eq. (12). In Eq. (14)we re-wrote the internal forces based on the final configuration, which is known. We can now treat Eq. (3), with the left hand side given by Eq. (14), the same way as in the case of a direct problem, and find the unknown displacements.

If some of the loading (included in the right hand side of the equation) is dependent on the un-deformed configuration, it must be re-written based on the deformed configuration in a similar manner. For example, body forces (due to applied accelerations or gravity) for each element can be written as:

$$ \mathbf{R}_b = \int_{V_0} {}_0\mathbf{H} \cdot \mathbf{b}^T dV_0 = \int_{V_f} {}_f\mathbf{H} \cdot \mathbf{b}^T \left| det\left(\frac{\partial_0\mathbf{x}}{\partial_f\mathbf{x}}\right)\right| dV_f = \int_{V_f} {}_f\mathbf{H} \cdot \mathbf{b}^T J^{-1} dV_f. \tag{15} $$

Note 1: If the shape functions ${}_f\mathbf{H}$ are linear over the elements of the discretization (and therefore the transformation φ, constructed using these shape functions, is linear over the elements), considering the relation between the discretizing nodes in the deformed and un-deformed configurations (Eq. (7)), the shape functions ${}_0\mathbf{H}$ are also linear.

Note 2: The exact numerical evaluation of the integrals that appear in Eqs. (14) and (15) can be performed only when constant strain elements with linear shape functions are used. Using higher order shape functions is also possible, but in such cases the functions to be integrated are not polynomials and therefore exact integration is not possible by using simple quadrature rules.

Given the above observation we will use constant strain elements (triangles/tetrahedrons with linear shape functions) for solving inverse deformation problems. When almost incompressible materials need to be modeled, the non-locking tetrahedral element presented in [18] can be used.

2.2. Implicit solution methods

The obtained equilibrium equation, which is based only on the known deformed configuration, can be solved using standard solution algorithms, such as Newton–Raphson iterations. Existing solution methods can be easily modified by replacing the subroutines for computing the stiffness matrix based on the internal forces defined by Eq. (4)with new subroutines based on the forces defined by Eq. (14).

The exact tangent stiffness matrix based on the definition of forces in Eq. (14)should be used for optimum convergence of the Newton–Raphson iterations. We will present the equations needed for the derivation of the exact tangent stiffness matrix in the following paragraphs.

The elements of the exact tangent stiffness matrix for each element can be defined as in [13]:

$$\mathbf{K}_{mnij} = \frac{\partial^f_0\mathbf{R}_{\mathbf{int}\ \ \mathrm{min}}}{\partial\mathbf{U}_{ij}}, \tag{16}$$

where $\mathbf{u}_{ij}$ is the displacement of node i of the element in the direction of coordinate j. In the following equations we will omit the subscripts wherever possible. Therefore, considering Eq. (14), we can re-write Eq. (16)as:

$$\mathbf{K} = \frac{\partial^f_0\mathbf{R}_{\mathbf{int}}}{\partial\mathbf{U}} = \int_{V_f}\frac{\partial}{\partial\mathbf{U}}\left({}_f\mathbf{D}\cdot{}^f_0\mathbf{F}\cdot{}^f_0\mathbf{S}^{\mathbf{T}}\cdot{}^f_0\mathbf{F}^{\mathbf{T}}J^{-1}\right)dV_f. \tag{17}$$

The derivative under the integral in Eq. (17)can be expanded as:

$$\frac{\partial}{\partial\mathbf{U}}\left({}_f\mathbf{D}\cdot{}^f_0\mathbf{F}\cdot{}^f_0\mathbf{S}^{\mathbf{T}}\cdot{}^f_0\mathbf{F}^{\mathbf{T}}J^{-1}\right) = {}_f\mathbf{D}\cdot\frac{\partial}{\partial\mathbf{U}}\left({}^f_0\mathbf{F}\cdot 0^{\mathbf{S}^{\mathbf{T}}}_f\cdot{}^f_0\mathbf{F}^{\mathbf{T}}J^{-1}\right), \tag{18}$$

$$\frac{\partial}{\partial\mathbf{U}}\left({}^f_0\mathbf{F}\cdot{}^f_0\mathbf{S}^{\mathbf{T}}\cdot{}^f_0\mathbf{F}^{\mathbf{T}}J^{-1}\right) = \frac{\partial^f_0\mathbf{F}}{\partial\mathbf{U}}{}^f_0\mathbf{S}^{\mathbf{T}}\cdot{}^f_0\mathbf{F}^{\mathbf{T}}J^{-1} + 0^{\mathbf{F}}_f\frac{\partial^f_0\mathbf{S}^{\mathbf{T}}}{\partial\mathbf{U}}{}^f_0\mathbf{F}^{\mathbf{T}}J^{-1} + {}^f_0\mathbf{F}\cdot{}^f_0\mathbf{S}^{\mathbf{T}}\frac{\partial^f_0\mathbf{F}^{\mathbf{T}}}{\partial\mathbf{U}}J^{-1} + {}^f_0\mathbf{F}\cdot{}^f_0\mathbf{S}^{\mathbf{T}}\cdot{}^f_0\mathbf{F}^{\mathbf{T}}\frac{\partial J^{-1}}{\partial\mathbf{U}}. \tag{19}$$

We will now expand each of the derivatives that appear in Eq. (19)(which defines, given the relationship between the Cauchy stress and the second Piola–Kirchhoff stress, the derivative of the Cauchy stress with respect to displacements). Based on the definition of the deformation gradient, given by Eq. (12), we can write:

$$\frac{\partial^f_0\mathbf{F}}{\partial\mathbf{U}} = \frac{\partial^0_f\mathbf{F}^{-\mathbf{1}}}{\partial\mathbf{U}} = -{}^0_f\mathbf{F}^{-\mathbf{1}}\frac{\partial^0_f\mathbf{F}}{\partial\mathbf{U}}{}^0_f\mathbf{F}^{-\mathbf{1}} = {}^0_f\mathbf{F}^{-\mathbf{1}}\frac{\partial({}_f\mathbf{U}^T\cdot{}_f\mathbf{D})}{\partial\mathbf{U}}{}^0_f\mathbf{F}^{-\mathbf{1}} = {}^f_0\mathbf{F}\frac{\partial({}_f\mathbf{U}^T\cdot{}_f\mathbf{D})}{\partial\mathbf{U}}{}^f_0\mathbf{F}, \tag{20}$$

with

$$\frac{\partial({}_f\mathbf{U}^T\cdot{}_f\mathbf{D})}{\partial\mathbf{U}_{ij}} = {}_f\mathbf{D}_{i:}\otimes\mathbf{e}_j, \tag{21}$$

where $\mathbf{D}_i$ represents line i of matrix $\mathbf{D}$, $\otimes$ represents the outer product and $\mathbf{e}_j$ is the versor corresponding to the degree of freedom j.

The derivative of the second Piola–Kirchhoff stress matrix can be computed based on the following equations:

$$\frac{\partial^f_0\hat{\mathbf{S}}}{\partial\mathbf{U}} = \frac{\partial^f_0\hat{\mathbf{S}}}{\partial^f_0\hat{\mathbf{E}}}\cdot\frac{\partial^f_0\hat{\mathbf{E}}}{\partial\mathbf{U}}, \tag{22}$$

where $\hat{\mathbf{S}}$ represents the vector form of the symmetric matrix $\mathbf{S}$, and the Green–Lagrange strain tensor ${}^f_0\mathbf{E}$ is defined as

$${}^f_0\mathbf{E} = \frac{1}{2}\left({}^f_0\mathbf{F}^{\mathbf{T}}\cdot{}^f_0\mathbf{F} - \mathbf{I}\right). \tag{23}$$

The matrix

$${}^f_0\mathbf{C} = \frac{\partial^f_0\mathbf{S}}{\partial^f_0\mathbf{E}}, \tag{24}$$

describes the material behavior, and is the same as in the case of direct problems. The derivative of the Green–Lagrange strain tensor is

$$\frac{\partial^f_0\mathbf{E}}{\partial\mathbf{U}} = \frac{1}{2}\left(\frac{\partial^f_0\mathbf{F}^{\mathbf{T}}}{\partial\mathbf{U}}\cdot{}^f_0\mathbf{F} + {}^f_0\mathbf{F}^{\mathbf{T}}\cdot\frac{\partial^f_0\mathbf{F}}{\partial\mathbf{U}}\right). \tag{25}$$

At last, the derivative of J^{-1} is calculated as:

$$\frac{\partial J^{-1}}{\partial \mathbf{U}} = \frac{\partial \det({}^f_0\mathbf{F}^{\mathbf{1}} -)}{\partial \mathbf{U}} = \frac{\partial \det({}^f_0\mathbf{F})}{\partial \mathbf{U}} = \left(\frac{\partial {}^f_0\mathbf{F}}{\partial \mathbf{U}} : {}^f_0\mathbf{F}^{-\mathbf{T}} \right) \cdot \det({}^f_0\mathbf{F}) = -\left(\frac{\partial ({}_f\mathbf{U}^T {}_f\mathbf{D})}{\partial \mathbf{U}} : {}^f_0\mathbf{F}^{\mathbf{T}} \right) \cdot J^{-1}, \tag{26}$$

where : represents the inner product between matrices.

We implemented the above solution method in an existing 3D implicit solver for solving direct elasticity problems. The main change required is the modification in the routine that computes the stiffness matrix at element level – the stiffness matrix is computed based on the definition of forces in Eq. (14)(and the above presented Eqs. (16)–(26)instead of Eq. (4). Another area that required changes is the storage of the stiffness matrix, as for the inverse procedure the stiffness matrix is no longer symmetric. This may also influence the choice of linear solver (a solver specialized for symmetric matrices can no longer be used).

2.3. Explicit solution methods

Alternative algorithms for obtaining the steady state solution can also be used. By adding the acceleration term and a mass proportional damping term to the left hand side of Eq. (3), a standard dynamics equation of motion is obtained, which can be solved using explicit time integration combined with Dynamic Relaxation:

$$\mathbf{M}^t_0\ddot{\mathbf{x}} + c\mathbf{M}^t_0\dot{\mathbf{x}} + {}^t_0\mathbf{R}_{\mathbf{int}} = {}^t\mathbf{R}_{ext}. \tag{27}$$

We modified such an algorithm, presented previously by us in [15], developed for solving direct elasticity problems. The only modification required was in the way the nodal forces for each element were computed. In the original code the nodal forces were computed using Eq. (3). In the modified code the computation was performed by combining Eqs. (12) and (14). The resulting solution algorithm was used for solving many of the inverse deformation problems presented in the next section.

The explicit integration algorithm used is only conditionally stable, with the stable integration step influenced by the maximum eigenvalue of the stiffness matrix. Because the modified explicit algorithm uses a different expression for force computation, the stiffness matrix will also be different. Nevertheless, we were able to use the same element based critical time step estimation as for solving direct problems in most cases that did not include excessive element distortions.

Because the purpose of a dynamic relaxation algorithm is to find the steady state solution, the mass matrix is chosen only based on considerations of stability and convergence speed, as it does not influence the steady state of the system [15]. Therefore, even if we start from a deformed configuration, there is no need to adapt the mass matrix to the changes in configuration, and the same treatment as in the case of solving direct problems is used. For details on the derivation of the solution method, including adaptive selection of the parameters, we direct the reader to reference [15].

The advantage of explicit methods over implicit ones depends on the size and type of the problem. A comparison between implicit and explicit methods, presented in [19], shows that the implicit method requires 4000 times more computations per time steps when compared to the explicit method for 3D problems. Due to their low memory requirements, explicit methods become even more attractive for problems with very large number of degrees of freedom, where the stiffness matrix cannot be stored in the fast core memory (RAM).

The presented explicit solution algorithm is very well suited for parallel implementation on graphics processing units (GPU). We implemented the explicit inverse deformation solution method on GPU following the same approach as for the direct solution method, due to the minimal differences between methods (computation of nodal forces). The details of the implementation are presented in [20].

3. Examples

In the following examples we used plane strain triangular elements for 2D simulations and constant strain tetrahedral elements for 3D simulations. We considered a Neo-Hookean material with the strain energy function:

$${}^t_0W = \mu_0(\bar{I}_1 - 3) + \frac{K_0}{2}(J-1)^2, \tag{28}$$

where $\bar{I}_1$ is the first invariant of the deviatoric Cauchy–Green strain tensor, μ_0 is the initial shear modulus and K_0 is the initial bulk modulus.

For most simulations, unless otherwise stated, we used a soft material with an initial Young's modulus of 3000 Pa and a Poisson's ratio of 0.49.

The first 4 examples were solved using the explicit inverse deformation procedure, while the last one was solved using the implicit inverse deformation procedure.

The general setup of the numerical experiments is as follows: we start with an initial known geometry, G, and loading. We then apply a direct procedure followed by the proposed inverse deformation procedure, or the inverse deformation procedure followed by a direct deformation procedure, and obtain a final geometry, F. We assessed the accuracy of the results by comparing the positions of the nodes of the initial geometry, G, with the positions corresponding to the final geometry, F. The difference in nodal positions compounds the errors from both the direct and the inverse deformation solutions.

For perfect solutions, there should be no difference in nodal positions. Numerical methods cannot provide a perfect solution, as there will always be errors generated by truncation, rounding, and convergence criteria used. Therefore, the inverse deformation solutions are proven to be correct as long as the obtained differences in nodal positions are deemed small. This is shown to be the case in all the following examples.

For the problems solved using the explicit method we used a large number of time steps in order to reduce the influence of the convergence criteria on the accuracy of the numerical solution.

3.1. Extension and compression of a rectangle

In this example we performed a constrained extension and compression of a rectangular domain by applying essential boundary conditions (displacing one of the faces while constraining the opposite face). We recovered the original shape of the rectangle using the explicit inverse deformation solution method, with the same essential boundary conditions. We used 2000 time steps for load application and 3000 time steps for finding the equilibrium position using Dynamic Relaxation, for both the direct and the inverse deformation solution algorithms. This example demonstrates the handling of very large deformations, with the rectangle being compressed by 35% of its length and extended by 50% of its length. The deformed and un-deformed configurations are presented in Fig. 1. The maximum obtained differences in nodal positions between the original and final geometries are presented in Table 1.

3.2. Gasket shape

In this example we consider the desired shape of a gasket while compressed and the distribution of pressure required at the interface (Fig. 2(a)). We then calculate the shape that should be manufactured (uncompressed) for two cases. In the first case there is high friction between the compressing surface and the gasket, simulated here, same as in [4], as a restriction on the lateral motion of the nodes found in contact with the surface (Fig. 2(b)). In the second case there is no friction between the gasket and the compressing surface (Fig. 2(c)).

Because of symmetry, only 1/4 of the gasket is actually modeled, and the symmetrical behavior is imposed by using suitable essential boundary conditions. This example demonstrates the correct handling of surface pressures and symmetries. We used 1000 time steps for load application and 2000 time steps for finding the equilibrium.

The required pressure is set to 400 kPa at the middle of the gasket, increasing linearly to 1200 kPa on the lateral quarter of the gasket. The resulting variation of the nodal forces at the interface is shown in Fig. 2(a).

The accuracy of the solution was checked by solving the direct problems and comparing the desired shape of the gasket with the computed one. For the direct problem solution the compression of the gasket was performed by moving the compressing surface and using a simple kinematic contact algorithm that prevented the interfacing nodes from penetrating the surface. This way we can also compare the resulting nodal forces with the required ones. The results of the analysis are presented in Table 2.

3.3. Beam under gravity loading

The purpose of this example is to demonstrate the capacity of the method to handle body forces (see Eq. (15)). A rectangular beam is fixed at one end and subjected to gravity loading. The un-deformed shape is then recovered using the proposed algorithm (see Fig. 3). We used 4000 time steps for load application and 6000 time steps for finding the equilibrium. The maximum difference in nodal positions between the un-deformed and the recovered geometries was 0.0042 mm for a beam of $4 \times 10\ \text{mm}^2$.

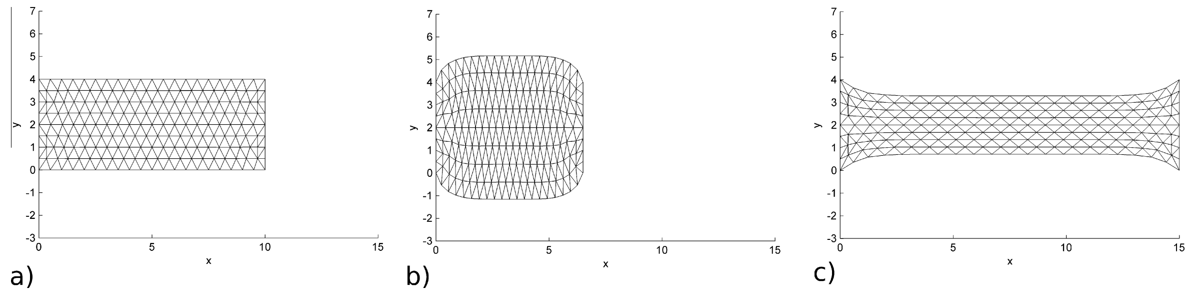


Fig. 1. The un-deformed/recovered (a), compressed (b), and extended geometry (c).

Table 1
Results from the rectangle extension and compression experiments.

Experiment	Rectangle size [mm^2]	Maximum applied deformation [mm]	Difference in nodal positions [mm]
Compression	4×10	3.5	3.8E−4
Extension	4×10	5	5.2E−5

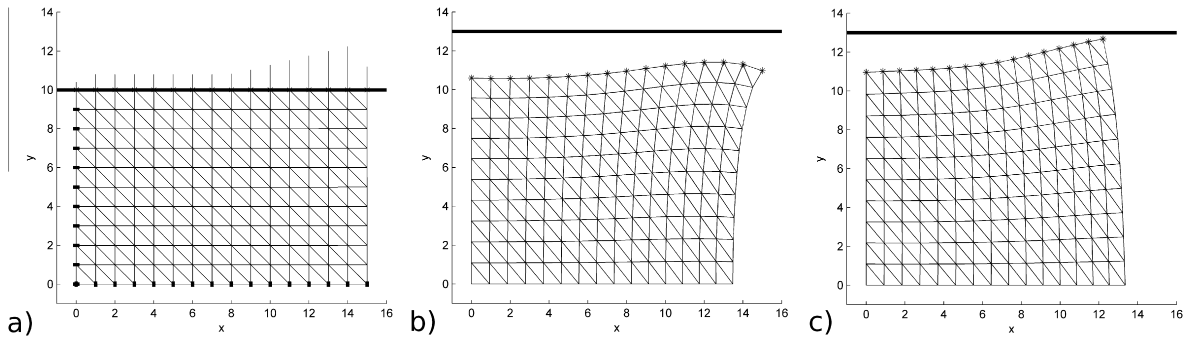


Fig. 2. (a) The deformed shape of the gasket (only 1/4 of the geometry shown, nodes that have essential boundary conditions applied due to symmetry are marked). The variation of the desired nodal forces on the compressed surface is also shown. (b) The un-deformed geometry with friction. (c) The un-deformed geometry with no friction.

Table 2
Results from the manufacturing gasket shape experiments.

Experiment	Gasket size [mm^2]	Difference in nodal forces [N]	Difference in nodal positions [mm]
Friction	30×20	0.0046	2.1E−6
No friction	30×20	0.0098	7.3E−6

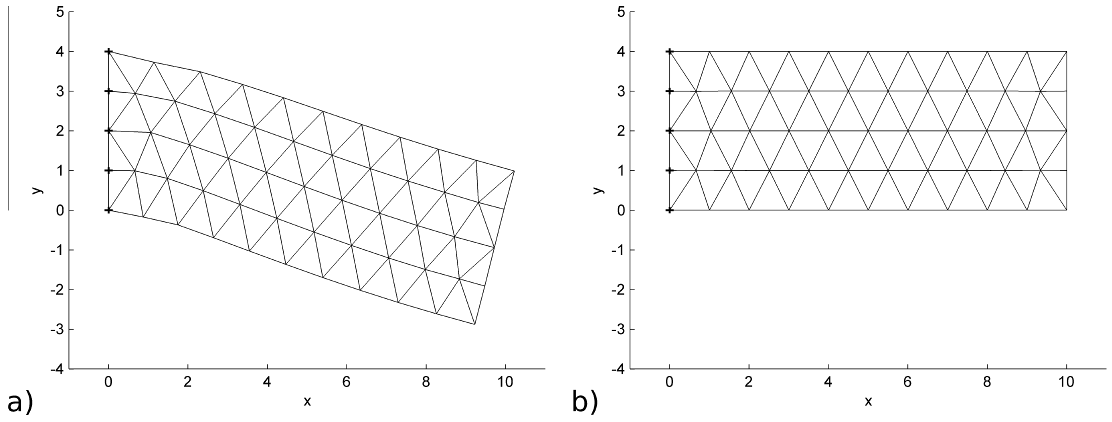


Fig. 3. (a) The deformed shape of the beam under gravity load and (b) The un-deformed/recovered geometry.

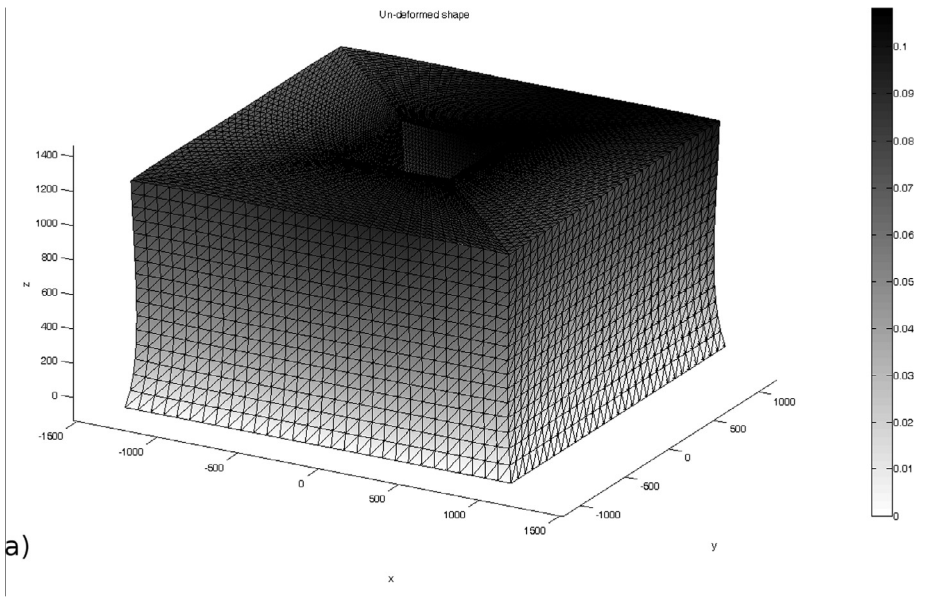
3.4. 3D object under gravity loading

This example demonstrates that our inverse deformation procedure implementation, including handling of gravity forces (whose value varies with the volume of the object) accurately computes the un-deformed configuration for 3D objects. It also demonstrates the efficiency of a parallel implementation of the explicit inverse deformation procedure on Graphics Processing Units (GPU).

To assess the accuracy of the method we performed the following experiment: for a chosen 3D geometry, considered as the deformed configuration under the effect of gravity, we applied our explicit inverse deformation procedure to compute the un-deformed configuration. The resulting un-deformed geometry was transferred into the commercial finite element software Abaqus [21], and loaded by gravity to compute the final geometry. The direct implicit solver with the default settings was used in Abaqus. We then compared the final geometry obtained from Abaqus with the starting deformed geometry for our inverse deformation procedure, defining the error as the difference at each nodal position (Fig. 4(b)). For solving the direct problem we also used our explicit dynamic relaxation procedure.

In order to increase the effect of gravity we considered a very soft and very heavy material, with a Poisson's ratio of 0.4. The resulting maximum displacement under gravity loading was 11% of the object's height. The maximum error relative to maximum displacements was 0.7% when the Abaqus implicit solver was used and 0.08% when our explicit dynamic relaxation solver was used.

In terms of efficiency of the parallel implementation on GPU, we compared the speed of the direct explicit solution method to the speed of the inverse deformation explicit solution method. For this problem, the mesh consisted of approximately 575 k elements and 100 k nodes (300 k degrees of freedom). The time required for the execution of 1000 explicit steps on GPU was 12.2 s for the inverse deformation method and 7.62 s for the direct method. Both methods executed more than 15 times faster on GPU than on the CPU and required approximately 5500 explicit steps to converge.



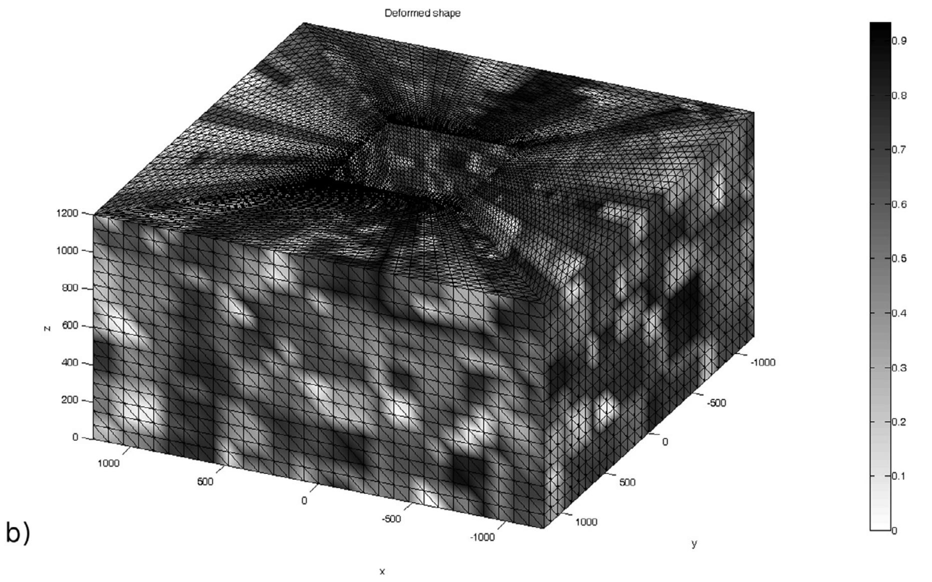


Fig. 4. (a) The un-deformed geometry of a 3D object. (b) The deformed geometry under gravity loading. Colors mark the difference in nodal positions between the geometry selected as the deformed geometry and the geometry recovered after applying our inverse deformation procedure, followed by a direct procedure. The direct procedure used was our explicit dynamic relaxation procedure in (a) and the Abaqus implicit procedure in (b).

We were unable to solve the problem using the implicit solution method due to insufficient RAM for storing the stiffness matrix, as our implementation cannot handle out of RAM matrix storage. The direct solution computation using the implicit Newton–Raphson method took 4 iterations and 2239 s in the finite element solver Abaqus, as compared to 630 s required by the explicit solution on the CPU and less than 40 s on the GPU. It is expected that an inverse deformation implicit solution method will take even longer to compute, due to the lack of symmetry of the stiffness matrix.

Computations were done on a PC with an Intel Core 2 Quad processor, 3 GB of RAM and Windows XP operating system. The GPU used was an NVIDIA Tesla C1060 having 4 GB of RAM and 240 cores. The newer generation GPUs, such as Tesla K40, can have up to 2880 cores, therefore promising even faster execution times.

3.5. Very large deformations in 3D

This example presents the deformation of a semi-circular rubber band with a square section resulting from the application of a constant force to its middle section (Fig. 5). This example showcases the use of implicit Newton–Raphson procedures for solving 3D problems involving very large deformations.

The rubber band was discretized using approximately 25 k tetrahedral elements and 6 k nodes.

Due to the very large resulting deformation, we applied the load in 2 steps and used a line search algorithm to maintain the convergence of the Newton–Raphson procedure. The linear solver used was MUMPS [22], which can handle both symmetric positive definite matrices (for direct problems) and general un-symmetric matrices (for inverse deformation problems).

We started with the un-deformed shape and applied the force to find the deformed shape. We then used the inverse deformation implicit procedure to recover the un-deformed shape. The maximum difference in nodal positions between the original and recovered un-deformed geometries was 1.1e−10.

In terms of displacement convergence of Newton–Raphson iterations, for this particular problem, the inverse deformation procedure shows much faster convergence as compared to the direct procedure. For an error limit of 1e−3, the direct procedure converges in 46 iterations while the inverse deformation procedure converges in 9 iterations (Fig. 6). The error for each iteration step is defined as the difference between the displacements calculated after that iteration and the displacements calculated after a large number of iterations (a maximum of 30 iterations per load step was configured). The use of the exact tangent stiffness matrix ensures good convergence properties for implicit inverse deformation solution methods.

4. Discussion and conclusions

In this paper we present a method for solving inverse deformation problems. The commonly used method for solving such problems involves the re-parameterization of the equilibrium equations in terms of the deformed configuration

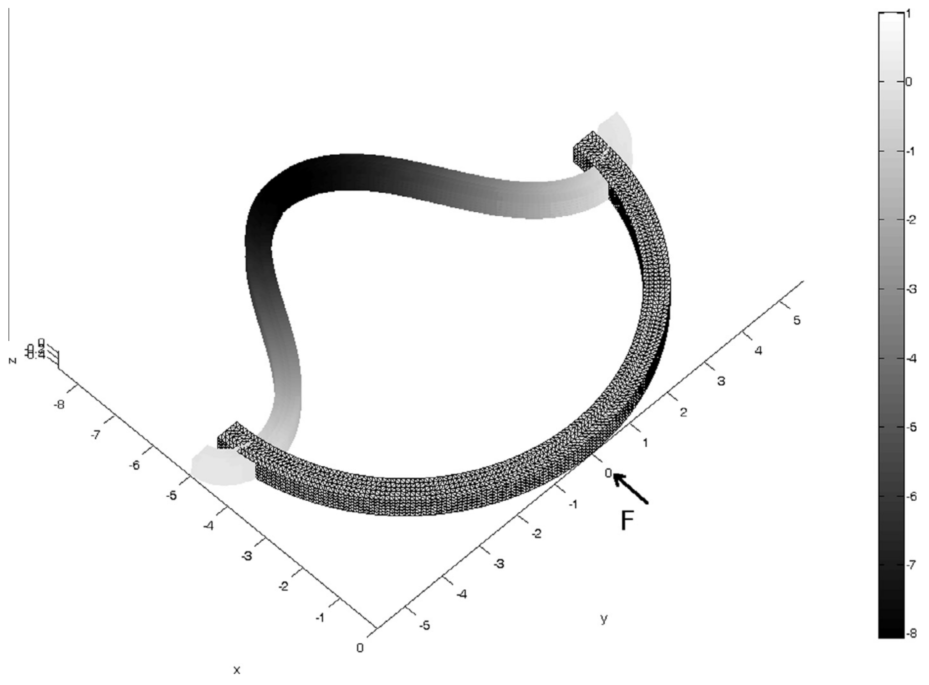
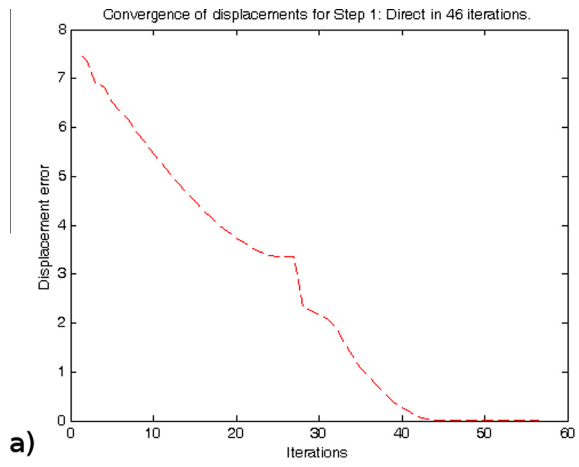


Fig. 5. Deformation of a rubber band under a force applied to its middle section. The un-deformed shape is shown as a mesh. The colors of the deformed shape mark the size of the resulting nodal displacements.



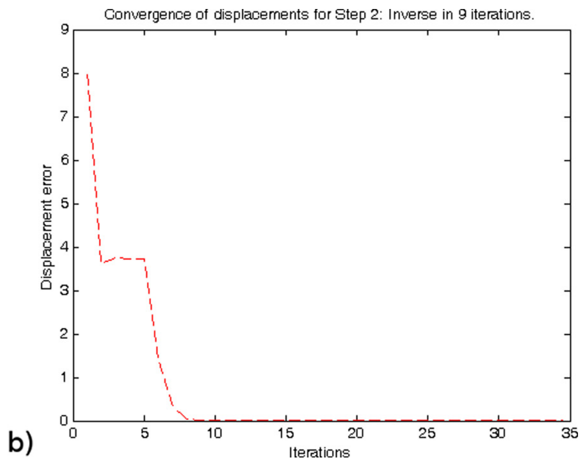


Fig. 6. Convergence in displacements for the direct (a) and inverse deformation (b) Newton–Raphson iterations.

(Eulerian description). Our method exploits the relationship between the un-deformed and the deformed configurations provided by the TL framework to rewrite the force equilibrium equation based on the known deformed configuration.

The proposed method has several advantages over the commonly used method presented by Govindjee and Mihalic in [4], especially when implemented in existing finite element software:

- It does not require the computation of the tangent operator describing the derivatives of the Cauchy stress to the inverse of the Cauchy–Green strain tensor ($D = \partial\sigma/\partial c$); existing material law implementations describing the relation between strain and stress can be used.
- The main change required for existing TL finite element software for solving direct problems is in the computation of stiffness matrix (implicit solution) or nodal forces (explicit solution) at element level.
- The explicit algorithm is very well suited for parallel implementation on Graphics Processing Units (GPU) [20,23], which can lead to significant speed-ups for highly non-linear problems with a large number of degrees of freedom.

We presented several examples of problems solved using the proposed method implemented in a TL explicit dynamics software which uses dynamic relaxation to find the static equilibrium solution. The examples demonstrate that the presented method can solve the inverse deformation problems under diverse loads (displacements, surface tractions and body forces). Although we were able to obtain results using the standard procedures for estimating the critical time step, we must mention that these procedures may not always work. The Jacobian of the system of equations describing the equilibrium is different from the one used in direct problems and it lacks symmetry (this is also the case for the method proposed in [4]). Computation of the critical time step for the inverse deformation explicit solution algorithm will require further investigation.

The presented explicit solution algorithm is very well suited for parallel implementation on graphics processing units (GPU). The presented example solved using our GPU implementation demonstrates the potential of greatly reducing the computation time for problems with a large number of degrees of freedom.

We developed the relations for the derivation of the stiffness matrix to be used with Newton–Raphson inverse deformation solution methods. We showed that such methods can be used to solve inverse deformation problems involving very large deformations at machine precision, therefore demonstrating the correctness of the presented approach.

Acknowledgements

The financial support of the Australian Research Council (Grant No. DP1092893 and DP120100402) is gratefully acknowledged.

References

[1] S. Govindjee, P.A. Mihalic, Computational methods for inverse deformations in quasi-incompressible finite elasticity, Int. J. Numer. Methods Eng. 43 (1998) 821–838.
[2] J. Lu, X. Zhou, M.L. Raghavan, Inverse elastostatic stress analysis in pre-deformed biological structures: demonstration using abdominal aortic aneurysms, J. Biomech. 40 (2007) 693–696.
[3] J. Lu, X. Zhou, M.L. Raghavan, Computational method of inverse elastostatics for anisotropic hyperelastic solids, Int. J. Numer. Methods Eng. 69 (2007) 1239–1261.
[4] S. Govindjee, P.A. Mihalic, Computational methods for inverse finite elastostatics, Comput. Methods Appl. Mech. Eng. 136 (1996) 47–57.

[5] V. Rajagopal, J.-H. Chung, D. Bullivant, P.M.F. Nielsen, M.P. Nash, Determining the finite elasticity reference state from a loaded configuration, Int. J. Numer. Methods Eng. 72 (2007) 1434–1451.
[6] K. Miller, J. Lu, On the prospect of patient-specific biomechanics without patient-specific properties of tissues, J. Mech. Behav. Biomed. Mater. 27 (2013) 154–166.
[7] F. Riveros, S. Chandra, E.A. Finol, T.C. Gasser, J.F. Rodriguez, A pull-back algorithm to determine the unloaded vascular geometry in anisotropic hyperelastic AAA passive mechanics, Ann. Biomed. Eng. 41 (2013) 694–708.
[8] A. Romo, P. Badel, A. Duprey, J.-P. Favre, S. Avril, In vitro analysis of localized aneurism rupture, J. Biomech. 47 (2014) 607–616.
[9] R.T. Shield, Inverse deformation results in finite elasticity, Z. Ange. Math. Phys. (ZAMP) 18 (1967) 490–500.
[10] P. Chadwick, Applications of an energy-momentum tensor in elastostatics, J. Elast. 5 (1975) 249–258.
[11] J.D. Eshelby, The elastic energy–momentum tensor, J. Elast. 5 (1975) 321–335.
[12] T. Yamada, Finite element procedure of initial shape determination for rubber-like materials, Technical Report No. 20, Res. Lab. Eng. Mat. Tokyo Institute of Technology, 1995.
[13] K.-J. Bathe, Finite Element Procedures, Prentice-Hall, New Jersey, 1996.
[14] K. Miller, G.R. Joldes, D. Lance, A. Wittek, Total lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation, Commun. Numer. Methods Eng. 23 (2007) 121–134.
[15] G.R. Joldes, A. Wittek, K. Miller, An adaptive dynamic relaxation method for solving nonlinear finite element problems. Application to brain shift estimation, Int. J. Numer. Method Biomed. Eng. 27 (2011) 173–185.
[16] G.R. Joldes, A. Wittek, K. Miller, Computation of intra-operative brain shift using dynamic relaxation, Comput. Methods Appl. Mech. Eng. 198 (2009) 3313–3320.
[17] T. Belytschko, K.W. Liu, B. Moran, Nonlinear Finite Elements for Continua and Structures, John Wiley & Sons Ltd, Chichester, 2006.
[18] G.R. Joldes, A. Wittek, K. Miller, Non-locking tetrahedral finite element for surgical simulation, Commun. Numer. Methods. Eng. 25 (2009) 827–836.
[19] T. Belytschko, A survey of numerical methods and computer programs for dynamic structural analysis, Nucl. Eng. Des. 37 (1976) 23–34.
[20] G.R. Joldes, A. Wittek, K. Miller, Real-time nonlinear finite element computations on GPU – application to neurosurgical simulation, Comput. Methods Appl. Mech. Eng. 199 (2010) 3305–3314.
[21] ABAQUS, ABAQUS Theory Manual Version 6.9, Dassault Systèmes Simulia Corp., Providence, RI, 2009.
[22] MUMPS home page, http://graal.ens-lyon.fr/MUMPS/, Last accessed: 04 March 2014.
[23] G.R. Joldes, A. Wittek, K. Miller, Real-time nonlinear finite element computations on gpu – handling of different element types, in: A. Wittek, P.M.F. Nielsen, K. Miller (Eds.), Computational Biomechanics for Medicine: Soft Tissues and Musculoskeletal System, Springer, New York, 2011, pp. 73–80.