Corotational cut finite element method for real-time surgical simulation: Application to needle insertion simulation

Huu Phuoc Bui, Satyendra Tomar, Stéphane Pierre Alain Bordas

▶ **To cite this version:**

Huu Phuoc Bui, Satyendra Tomar, Stéphane Pierre Alain Bordas. Corotational cut finite element method for real-time surgical simulation: Application to needle insertion simulation. Computer Methods in Applied Mechanics and Engineering, 2018, 10.1016/j.cma.2018.10.023 . hal-01921631

HAL Id: hal-01921631
https://hal.science/hal-01921631

Submitted on 13 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Corotational cut finite element method for real-time surgical simulation: Application to needle insertion simulation

Huu Phuoc Bui[a,b,*], Satyendra Tomar[a], Stéphane P.A. Bordas[a,c,*]

[a] *Institute of Computational Engineering, University of Luxembourg, Faculty of Sciences Communication and Technology, Luxembourg*
[b] *Laboratoire de Mathématiques de Besançon (LMB) UMR CNRS 6623, Université de Franche-Comté, Besançon, France*
[c] *Institute of Mechanics and Advanced Materials, School of Engineering, Cardiff University, UK*

Abstract

We present the corotational cut Finite Element Method (FEM) for real-time surgical simulation. The only requirement of the proposed method is a background mesh, which is not necessarily conforming to the boundaries/interfaces of the simulated object. The details of the surface, which can be directly obtained from binary images, are taken into account by a multilevel embedding algorithm which is applied to elements of the background mesh that are cut by the surface. Dirichlet boundary conditions can be implicitly imposed on the surface using Lagrange multipliers, whereas traction or Neumann boundary conditions, which is/are applied on parts of the surface, can be distributed to the background nodes using shape functions. The implementation is verified by convergences studies, of the geometry and of numerical solutions, which exhibit optimal rates. To verify the reliability of the method, it is applied to various needle insertion simulations (e.g. for biopsy or brachytherapy) into brain and liver models. The numerical results show that, while retaining the

*Corresponding authors:
Email addresses: huu-phuoc.bui@alumni.unistra.fr (Huu Phuoc Bui), stephane.bordas@alum.northwestern.edu (Stéphane P.A. Bordas)

Preprint submitted to Elsevier

accuracy of the standard FEM, the proposed method can (1) make the discretisation independent from geometric description, (2) avoid the complexity of mesh generation for complex geometries, and (3) provide computational speed suitable for real-time simulations. Thereby, the proposed method is very suitable for patient-specific simulations as it improves the simulation accuracy by automatically, and properly, taking the simulated geometry into account, while keeping the low computational cost.

Keywords:
Cut finite element method, Unfitted FEM, Corotational CutFEM, Needle insertion, Real-time simulation

---

1. Introduction

Nowadays real-time simulation plays an important role in different fields: graphic animation [1], fracture of stiff materials [2], and surgical training and simulation [3, 4, 5, 6], to name a few. In the medical context, surgical simulations are not only useful for training, but also helpful for pre-operative planning, and intra-operative guidance. Surgical simulations have to take into account interactions between a surgeon or an interventional radiologist with a deformable organ via surgical instruments (e.g. a needle), and also interactions between the organ with its neighbouring structures. To be useful in real-life situations, it is required that the computations are performed in real-time. To achieve real-time performance, some advanced solvers, e.g. GPU-based computation [7], or asynchronous solver [6], can be used. The computational time can also be reduced by using coarse meshes. However, using coarse meshes, some geometric details are often lost. Simulations using coarse meshes are thus only suitable for targeted surgical training which relies more on visual reality than on actual one. Such simulations fall short for surgical planning or guidance where computations must provide accurate results. A model order reduction technique can also be used to solve system equations with significantly less time but it results in lower accuracy [8]. To

reduce the computational effort, Quesada et al [9] propose a computational parametric meta-model which is computed offline, and is only evaluated online.

Medical simulations have to deal with complex anatomical structures, e.g. prostate, blood vessel, liver, brain, brain ventricle, etc. When the patient-specific geometry is considered, the mesh of the organ needs to be reconstructed since the organ geometry is different from one patient to another. Misra and coworkers [10] have shown that the geometry of the organ and boundary conditions surrounding the organ are the most important factors influencing the organ deformation, and thus have a direct impact on the accuracy of simulation and planning. And when supercomputers or parallel computation is not considered, in order to response to real-time simulation requirements, computations involving interactions with surgical tools and/or cutting operations are performed on coarse meshes while applying pre-computed deformation from fine meshes, see e.g., [4]. However, these coarse and fine meshes may not conform to each other, resulting in different geometric descriptions and different boundary conditions, and thus the question of simulation accuracy demands further investigation.

The use of geometric description in a computational method, so that the users only need to provide a background mesh which is not necessarily conforming to the boundary geometry of the simulated object, can dramatically reduce computational cost of preprocessing. This is the idea behind the fictitious domain method [11, 12], or the cut finite element method (CutFEM) [13]. The extended finite element method (XFEM) [14, 15], and the generalised finite element method (GFEM) [16] have been developed to deal with crack surfaces, or material interfaces evolving during simulation. These approaches allow for computation of coupled physical processes on distinct subregions of the total volume, and do not require an absolute conformity between the meshes from mesh construction as in [17, 18]. In the context of real-time patient-specific surgical simulations, using a unified geometric

description in a finite element code can help in (1) reducing preprocessing cost of mesh generation, and (2) implicitly improving accuracy of simulation and planning (compared to the situation in which coarse meshes are employed in an inappropriate manner to achieve real-time requirements).

The main contribution of our paper is to propose the corotational CutFEM approach which is suitable for real-time patient specific simulations. The corotational model [19] is widely used for the treatment of large rotations of soft tissues, see e.g., [7, 20, 21, 22, 23, 24]. By using the proposed methodology, geometries of simulated objects can be captured automatically. We also show that, for a given accuracy, using the proposed method is computationally more advantageous than employing the standard FEM. On the implementation aspect of the method, we propose the so-called multilevel embedding approach to correctly integrate implicit boundaries of the simulated organ, and to accurately capture implicit interfaces of e.g., a tumour. The implementation is verified by convergence studies, which exhibit optimal rates. We use Lagrange multipliers to impose Dirichlet boundary conditions on implicit boundaries, whereas traction or Neumann boundary conditions, which is/are applied on parts of the surface, can be distributed to the background nodes using shape functions. We demonstrate the performance of the corotational CutFEM through various applications: from needle insertion simulations (e.g. for biopsy or brachytherapy) to simulation of electrode lead implantation in Deep Brain Stimulation (DBS) procedure. The algorithm is implemented in the open-source SOFA framework [25] [1].

The remaining of the paper is organised as follows. In Section 2, we describe the formulation of the needle insertion problem into soft tissues, together with its discrete form. We then present the geometry discretisation of cut elements, and the algorithm for multilevel embedding approach to correctly integrate implicit boundaries/interfaces. It is then followed by the discussion

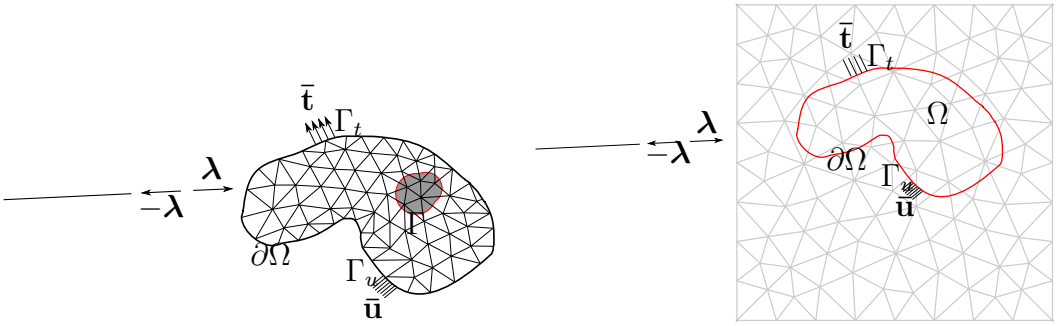
[1] https://www.sofa-framework.org

on implementation aspects. Next, a corotational formulation for CutFEM is briefly introduced. It is followed by the description of how boundary conditions are applied on implicit surfaces. This section is concluded with the discussion on solving the system equations with constraints. Numerical results are presented in Section 3, which demonstrate the capabilities of the approach through various needle insertion problems into liver and brain models. Finally, conclusions are drawn in Section 4.

2. Methods

2.1. Problem setting

In the context of needle insertion into soft tissue, we model both, the needle and the tissue, as dynamic deformable objects. Creating a good quality conforming mesh, when the tissue is modelled as heterogeneous material with complex internal structures, or when the tissue has a complex geometry, is a very complicated and challenging task. To overcome this burden, CutFEM has been proposed as it does not require conforming meshes. Figure 1a schematically shows a problem in which an interface is immersed into a tissue geometry for simulating e.g. a tumour geometry. Figure 1b shows a problem in which the tissue is simulated with an implicit boundary (i.e. the computational mesh is not fitted to the tissue geometry).



(a) Immersed interface problem. (b) Implicit boundary problem.

Figure 1: Two dimensional representation of the problems studied in the paper.

Let Ω represent the tissue domain, and $\partial\Omega$ denote its boundary. The tissue undergoes an imposed displacement $\bar{\mathbf{u}}$ on the boundary part Γ_u, and a traction force $\bar{\mathbf{t}}$ on the boundary part Γ_t. The governing equation of the problem reads

$$\operatorname{div}\boldsymbol{\sigma} + \hat{\boldsymbol{b}} + \boldsymbol{\lambda} = \rho\ddot{\boldsymbol{u}} \quad \text{in } \Omega, \tag{1a}$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \bar{\mathbf{t}} \quad \text{on } \Gamma_t, \tag{1b}$$

$$\boldsymbol{u} = \bar{\mathbf{u}} \quad \text{on } \Gamma_u, \tag{1c}$$

where $\boldsymbol{u}$ is the displacement field of the object, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\hat{\boldsymbol{b}}$ is the generalised body force vector, ρ is the mass density, $\ddot{\boldsymbol{u}}$ is the second partial derivative of $\boldsymbol{u}$ with respect to time, $\boldsymbol{n}$ denotes the outward unit normal vector on Γ_t, and $\boldsymbol{\lambda}$ denotes the interaction force between the needle and the tissue. Within the dynamic behaviour consideration, the generalised body force $\hat{\boldsymbol{b}}$ is expressed as

$$\hat{\boldsymbol{b}} = \bar{\boldsymbol{b}} - c\dot{\boldsymbol{u}}, \tag{2}$$

where $\bar{\boldsymbol{b}}$ is the body force per unit volume, $-c\dot{\boldsymbol{u}}$ expresses the resistance opposite to the motion velocity $\dot{\boldsymbol{u}}$, and c is the viscosity parameter which can be considered to be numerically given value [26], see Section 2.2.

The strain, which is a deformation measure, is expressed through the displacement as

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right). \tag{3}$$

In this paper, we employ a linear elastic material based on corotational formulation, see also Section 2.4, so that the stress-strain relationship is expressed as

$$\boldsymbol{\sigma} = \lambda\operatorname{div}(\boldsymbol{u})\mathbf{I} + 2\mu\boldsymbol{\epsilon}, \tag{4}$$

where λ and μ are the Lamé's coefficients, and $\mathbf{I}$ denotes the identity matrix. The Lamé's coefficients are computed from the Young's modulus E and

Poisson's ratio ν as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{5}$$

The interaction force $\boldsymbol{\lambda}$ is defined from the interaction law between the needle and the tissue. Three type of constraints between the needle and the tissue are defined during needle insertion simulations: constraint for puncturing at tissue surface, constraint at needle tip, and constraints along the needle shaft, see [27, 28] for details. For the sake of completeness, these constraints are recalled here. When the needle has not been penetrated into the tissue yet, the contact between them verifies the Kuhn-Tucker condition in the direction n (normal to tissue surface)

$$\delta_n \geq 0; \quad \lambda_n^{ts} \geq 0; \quad \delta_n \cdot \lambda_n^{ts} = 0, \tag{6}$$

where λ_n^{ts} stands for the normal component of the constraint on the tissue surface, δ_n denote the distance between the needle tip and the tissue surface in the direction n. This condition expresses the fact that the contact force only exists when the needle tip is in contact with the tissue surface ($\delta_n = 0$). In the tangential direction, a frictional contact is expressed through the Coulomb's friction law

$$\lambda_t^{ts} < \mu_f \lambda_n^{ts} \quad \text{(stick)}; \quad \lambda_t^{ts} = \mu_f \lambda_n^{ts} \quad \text{(slip)}, \tag{7}$$

with μ_f denoting the coefficient of friction. When the normal component of the constraint is greater than the tissue puncture strength λ_{p_0}, which is read

$$\lambda_n^{ts} > \lambda_{p_0}, \tag{8}$$

the needle tip can penetrate into the tissue.

Once inside the tissue, at the needle tip, a constraint is defined between the latter and the tissue. This constraint expresses the phenomenon that the tip is stuck, or can cut and advance inside the tissue. The condition for this

constraint reads

$$\lambda_n^{nt} < \mu\lambda_t^{nt} + \lambda_{c_0} \text{ (stick)}; \quad \lambda_n^{nt} \geq \mu\lambda_t^{nt} + \lambda_{c_0} \text{ (cut and slip)}, \tag{9}$$

where n now denotes the normal direction which is the direction of the needle shaft at the tip, the superscript nt indicates the constraint at the needle tip, λ_{c_0} is the cutting strength of the tissue.

Since the needle shaft should follow the trajectory created inside the tissue by the needle tip when the latter advances, constraints along the trajectory are defined between the tissue and the needle shaft which fulfil the Coulomb's friction law

$$\lambda_n^{ns} < \mu\lambda_t^{ns} \text{ (stick)}; \quad \lambda_n^{ns} = \mu\lambda_t^{ns} \text{ (slide)}, \tag{10}$$

where the superscript ns indicates the constraints along the needle shaft.

These three types of constraint are schematically illustrated in Figure 2. It is noted that the total interaction force between the needle and the tissue, at each time step, is nothing different from the normal force f_n computed at a cross-section of the needle model which has not been penetrated into the tissue yet, as illustrated in Figure 2.

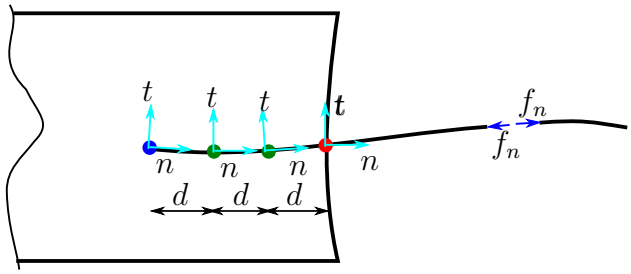


Figure 2: Schematic representation of constraint at the tissue surface $\boldsymbol{\lambda}^{ts}$ (shown by red colour), constraints defined along needle shaft $\boldsymbol{\lambda}^{ns}$ (illustrated by green colour), and constraint at the needle tip $\boldsymbol{\lambda}^{nt}$ (depicted by blue colour). These constraints are defined on a corresponding local frame nt, and they are located at the corresponding constraint points separated by a distance d. The interaction force between the needle and the tissue is simply represented by the normal force f_n of the needle model, computed at the non-penetrated (according to the tissue) cross-section of the needle shaft.

2.2. Weak form and FEM discretisation

For the numerical studies in the following, we employ homogeneous boundary condition on the Dirichlet boundary Γ_u i.e. $\bar{\boldsymbol{u}} = \boldsymbol{0}$. We denote the Sobolev space in three dimensions as $(H^1(\Omega))^3$, and then define the space of admissible displacements $\boldsymbol{V}$ as

$$\boldsymbol{V} = \{\boldsymbol{v} \in (H^1(\Omega))^3 : \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_u\}. \tag{11}$$

To begin with, for the sake of simplicity, we omit the interaction force $\boldsymbol{\lambda}$ in Equation (1a), and only consider the tissue/needle without any interaction with the other. The constraints for the interaction force will be dealt later. By multiplying Equation (1a) by a test function $\boldsymbol{v} \in \boldsymbol{V}$, and integrating by parts on the domain Ω, we obtain

$$\int_\Omega \mathrm{div}\boldsymbol{\sigma} \cdot \boldsymbol{v} \, \mathrm{d}\Omega + \int_\Omega \boldsymbol{b} \cdot \boldsymbol{v} \, \mathrm{d}\Omega = \int_\Omega \rho \ddot{\boldsymbol{u}} \cdot \boldsymbol{v} \, \mathrm{d}\Omega. \tag{12}$$

After performing the integration by parts to the integrand of the first term of the left-hand side in Equation (12), and noting that $\boldsymbol{v}$ vanishes on the boundary part Γ_D, we end up with this variational formulation

$$\begin{cases} \text{Find a displacement } \boldsymbol{u} \in \boldsymbol{V} \text{ such that, } \forall \boldsymbol{v} \in \boldsymbol{V}, \\ \underbrace{\displaystyle\int_\Omega \rho \ddot{\boldsymbol{u}} \cdot \boldsymbol{v} \, \mathrm{d}\Omega + \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \, \mathrm{d}\Omega + \int_\Omega c\dot{\boldsymbol{u}} \cdot \boldsymbol{v} \, \mathrm{d}\Omega}_{q(\boldsymbol{u},\boldsymbol{v})} = \underbrace{\displaystyle\int_\Omega \bar{\boldsymbol{b}} \cdot \boldsymbol{v} \, \mathrm{d}\Omega + \int_{\Gamma_t} \bar{\boldsymbol{t}} \cdot \boldsymbol{v} \, \mathrm{d}\Gamma}_{l(\boldsymbol{v})} \end{cases} \tag{13}$$

To solve the problem described by Equation (13) numerically, we need to transform the continuous variational problem (13) to a discrete variational one. This is performed by introducing a *finite-dimensional space*, denoted by $\boldsymbol{V}_h \subset \boldsymbol{V}$. It is noted that in the CutFEM framework, for tissue deformation simulations, the space $\boldsymbol{V}_h$ is defined on the mesh obtained from elements located inside the tissue surface and from elements cut by the tissue surface,

see Section 2.3. The problem now reads

$$\begin{cases} \text{Find a displacement } \boldsymbol{u}_h \in \boldsymbol{V}_h \text{ such that, } \forall \boldsymbol{v}_h \in \boldsymbol{V}_h, \\ q(\boldsymbol{u}_h, \boldsymbol{v}_h) = l(\boldsymbol{v}_h). \end{cases} \tag{14}$$

The space $\boldsymbol{V}_h$ is constructed from a mesh Ω_h obtained by spatially discretising the domain Ω into n_e finite elements Ω_e, $e = 1, 2, \ldots, n_e$, using n_n nodes. For each element Ω_e, we define an interpolation function (called shape function), denoted by $\boldsymbol{N}^e = [N_j^e]$, $j = 1, 2, \ldots, m_e$ with m_e the number of nodes of the element e. Mathematically, these are linear (P^1) elements in three-dimensions. By integrating every term of Equation (14), and then assembling for the whole domain Ω_h, we obtain the following discrete problem, in which the subscript h is dropped for the sake of simplicity (see, e.g., [29, 30])

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{C}\dot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}^{ext}, \tag{15}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{C}$ is the damping matrix, and $\boldsymbol{f}^{ext}$ is the external force vector of the whole system. These quantities are assembled from the corresponding element quantities: $\boldsymbol{M}^e = \int_{\Omega_e} \boldsymbol{N}^{eT} \rho \boldsymbol{N}^e \, \mathrm{d}\Omega$, $\boldsymbol{C}^e = \int_{\Omega_e} \boldsymbol{N}^{eT} c \boldsymbol{N}^e \, \mathrm{d}\Omega$, $\boldsymbol{K}^e = \int_{\Omega_e} \boldsymbol{B}^{eT} \boldsymbol{E} \boldsymbol{B}^e \, \mathrm{d}\Omega$, in which $\boldsymbol{B}^e$ is the strain-displacement matrix computed from the derivative of the shape function $\boldsymbol{N}^e$, and $\boldsymbol{E}$ is the fourth order tensor describing the stress-strain relationship $\boldsymbol{\sigma} = \boldsymbol{E} : \boldsymbol{\varepsilon}$. Moreover, $\boldsymbol{f}^{eext}$ is computed from the body force $\bar{\boldsymbol{b}}$ and the traction $\bar{\boldsymbol{t}}$ acting on the element e. In this paper, a lumped mass matrix is employed, where a diagonal mass matrix (from the mass density ρ) is integrated over the volume of each element. The stiffness matrix $\boldsymbol{K}$ is computed using the corotational FEM for both, the needle and the tissue, see Section 2.4. In practice, it is difficult to determine the damping matrix $\boldsymbol{C}$ as knowledge of viscous parameter c is lacking. In our simulation, we employ Rayleigh damping which is a linear combination of stiffness and mass

matrices, see [31]

$$\boldsymbol{C} = \alpha \boldsymbol{M} + \beta \boldsymbol{K}. \tag{16}$$

We set $\alpha = 0.01$, and $\beta = 0.1$ in our simulations.

The relation (15) can be rewritten as

$$\boldsymbol{M}\mathbf{a} = \boldsymbol{f}(\mathbf{x}, \mathbf{v}), \tag{17}$$

where $\mathbf{x}$, $\mathbf{v} = \dot{\mathbf{u}}$, and $\mathbf{a} = \ddot{\boldsymbol{u}}$, represent the position, velocity, and acceleration, respectively, and $\boldsymbol{f}(\mathbf{x}, \mathbf{v}) = \boldsymbol{f}^{ext} - \boldsymbol{K}\boldsymbol{u} - \boldsymbol{C}\mathbf{v}$ represents the net force (the difference of the external and internal forces) applied to the object. It is noted that we will use corotational formulation, see [32], to better capture the tissue/needle deformation under large rotational deformation, see also [33], which often occurs in surgical simulations. Therefore, the system term $\boldsymbol{K}\boldsymbol{u}$ is computed from the contribution of each element as will be described in Section 2.4.

For temporal discretisation of Equation (17), i.e. to numerically solve the problem in time, an implicit backward Euler scheme [34] is used to discretize the velocity and position as follows

$$\dot{\boldsymbol{u}}_{t+\tau} = \dot{\boldsymbol{u}}_t + \tau \ddot{\boldsymbol{u}}_{t+\tau}; \quad \boldsymbol{u}_{t+\tau} = \boldsymbol{u}_t + \tau \dot{\boldsymbol{u}}_{t+\tau}, \tag{18}$$

where τ denotes the time step. Inserting Equation (18) into Equation (17) yields the final discrete system

$$\underbrace{(\boldsymbol{M} + \tau \boldsymbol{C} + \tau^2 \boldsymbol{K})}_{\mathbf{A}} d\mathbf{v} = \underbrace{\tau \boldsymbol{f}(\mathbf{x}^t, \mathbf{v}^t) - \tau^2 \boldsymbol{K}\mathbf{v}^t}_{\boldsymbol{b}} \tag{19}$$

or simply $\mathbf{A}d\mathbf{v} = \boldsymbol{b}$, where $d\mathbf{v} = \mathbf{v}_{t+\tau} - \mathbf{v}_t$. Note that here we use $\boldsymbol{b}$ to denote the right-hand side of Equation (19), and it does not mean the generalised body force described in Equation (1a).

When taking into account the interaction between the needle and the

tissue, Equation (19) is rewritten as

$$\mathbf{A}d\mathbf{v} = \boldsymbol{b} + \boldsymbol{H}^T\boldsymbol{\lambda}, \tag{20}$$

in which $\boldsymbol{H}^T$ provides the direction of the constraints. The interaction constraint $\boldsymbol{\lambda}$, between the needle and the tissue, is computed using Lagrange multipliers, see Section 2.6. To understand the matrix $\boldsymbol{H}$, we suppose that the needle is going to have contact with the tissue surface at the point P, as illustrated in Figure 3. When they have been in contact, the relative displacement between the needle tip Q and the contact point P in the direction of the contact $\boldsymbol{n}$ vanishes

$$\delta_n = (\boldsymbol{u}_P - \boldsymbol{u}_Q) \cdot \boldsymbol{n} = 0. \tag{21}$$

Furthermore, the displacement at the contact point P on the tissue surface can be interpolated from the nodal displacements of the corresponding triangle. Also, the displacement of the needle tip can be interpolated from the corresponding needle element (1D element embedded in 3D space, described later)

$$\boldsymbol{u}_P = \sum_i N_i^1(P)\boldsymbol{u}_i; \quad \boldsymbol{u}_Q = \sum_i N_i^2(Q)\boldsymbol{u}_i, \tag{22}$$

where N_i^1, and N_i^2 are the shape functions defined for the elements of the tissue surface and the needle, respectively. By introducing Equation (22) into Equation (21), we can write

$$\boldsymbol{H}^1\boldsymbol{u}^1 - \boldsymbol{H}^2\boldsymbol{u}^2 = 0, \tag{23}$$

in which $\boldsymbol{u}^1$, and $\boldsymbol{u}^2$ are the nodes involving due to contact of the tissue surface and of the needle, respectively. By a similar approach, we can write the interaction force due to contact applied to the involving nodes on the

tissue and the needle as

$$\boldsymbol{H}^{1^T}\lambda = -\boldsymbol{H}^{2^T}\lambda, \tag{24}$$

with λ the interaction force in the contact space. It is noted that, when using CutFEM for simulation of soft tissue, the tissue surface is not conforming to the computational mesh, the force $\boldsymbol{H}^{1^T}\lambda$ is again interpolated into the involving nodes of the computational mesh, see also Section 2.5.

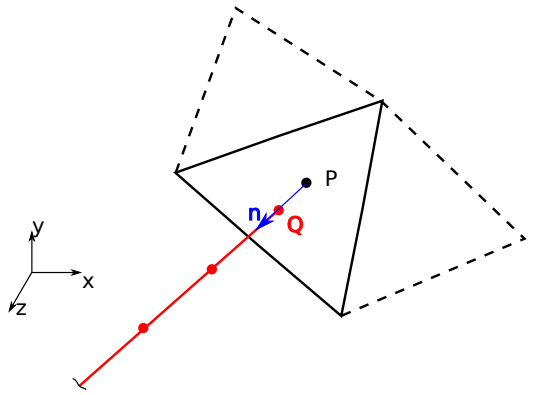


Figure 3: Schematic representation of contact between the needle (shown by red colour) and the tissue surface (shown by black colour).

After solving (20) for $d\mathbf{v}$, the position and velocity for needle and tissue are updated as

$$\mathbf{v}_{t+\tau} = d\mathbf{v} + \mathbf{v}_t; \quad \mathbf{x}_{t+\tau} = \mathbf{x}_t + \tau\mathbf{v}_{t+\tau}. \tag{25}$$

The tissue domain is discretised by a tetrahedron mesh. For the tetrahedra that intersect with the immersed interface/implicit boundary, an embedded element set is employed to facilitate the integration, see Section 2.3.

Since the length of the needle is much greater than the dimensions of its cross section, it can be safely assumed that the cross section, which is normal to the mid-line (i.e. the neutral line) of the needle, remain planar and normal to the mid-line during deformation. This makes it possible to use the Euler-Bernoulli beam theory [35] to describe the behaviour of the needle. The

needle is then discretised by one dimensional elements, and suitable shape functions (C^1 continuous, e.g. Hermite) are employed for the interpolation of the displacement field so that the bending behaviour of the needle is well taken into account. By this, each node of the elements used for the needle has 6 degrees of freedom (3 translations and 3 rotations).

2.3. Geometry discretisation for immersed/implicit boundaries

The discretisation technique presented here is applicable for both, fictitious domain problems [11, 36] and nonconforming interface problems, see, e.g. [37].

2.3.1. Geometry discretisation

In fictitious domain or nonconforming interface approaches, the boundary of a given domain is embedded into a computational mesh. The latter is used to approximate the solution of the governing PDEs. In general, the boundary surface of the given domain is represented by a very fine mesh, and the boundary surface mesh and the computational mesh are not assumed to be conforming. In fictitious domain method, the governing equations are integrated only on the inside volume (i.e. Ω_2, see Figure 4a) bounded by the surface boundary, whereas in interface problems, during integration, different mechanical properties are assigned to the outside volume Ω_1 and inside volume Ω_2, which are separated by the surface, see Figure 4a. The domain is discretised by a mesh which is nonconforming with respect to the surface Γ, as schematically shown in Figure 4b.

In order to accurately integrate on a given domain, the elements are firstly classified into three categories: cut elements, inside elements (i.e. inside of Γ) and outside elements (i.e. outside of Γ), see Figure 4c. To facilitate integration, each cut element is then embedded with a sub-element set consisting of elements which are conforming with the surface Γ. It is important to note that the sub-element set is only used for integration purpose, and since the degrees of freedom of the cut elements are still defined only

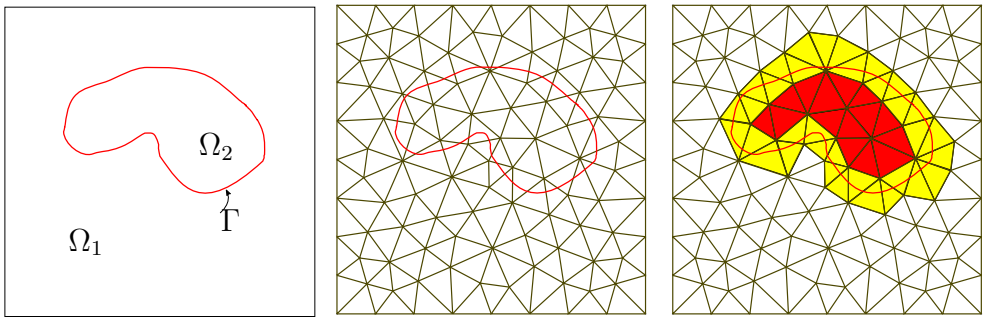


Figure 4: Two dimensional representation of the problem domains (a), the domains are nonconformingly discretised with respect to the surface Γ (b), the cut elements are highlighted by yellow colour whereas all inside elements of the volume Ω_2 are highlighted by red colour, the remaining elements are marked as outside (i.e. outside of Γ) elements (c).

on their nodes, the sub-element embedding approach does not affect the approximation properties of the discretisation at all.

To identify the cut elements, we use the level-set method [38]. Figure 5 schematically shows an element which is potentially cut by the interface. To know if the element is cut by the interface, we define the level-set function as a signed distance function from the nodes of the element to the interface, and check the sign of that function. An edge P_iP_j of the element is cut by the interface if and only if

$$\delta(P_i) \cdot \delta(P_j) < 0, \tag{26}$$

where P_i and P_j denote the position of the two ends of the edge, and δ is the signed distance defined from the points to the interface, see Figure 5. The intersection between the interface and an edge is approximated by the zero level-set. When Equation (26) is fulfilled, the barycentric coordinate of the intersection reads

$$\xi = \frac{|\delta(P_i)|}{|\delta(P_i)| + |\delta P_j|}, \tag{27}$$

and the position of the intersection is computed as

$$P = (1 - \xi)P_i + \xi P_j. \tag{28}$$

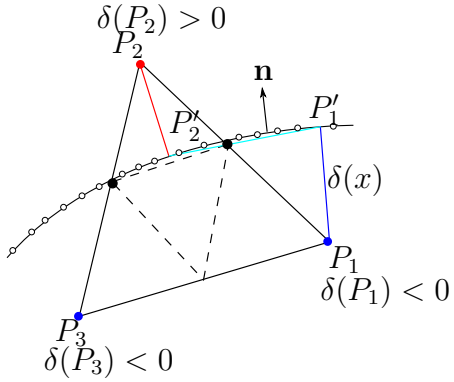


Figure 5: Schematic representation of intersection computation between a triangle and a discretised surface. An embedded subtriangulation is performed on the element for integration purpose.

In general applications, the immersed interface has a complex geometry, and it is not necessarily a convex surface, see Figure 9a for a liver surface as an example. Therefore, using the level set function to identify the inside and outside elements may not be efficient, especially when the elements are far from the interface. In Section 2.3.4, an efficient algorithm is presented to identify inside and outside elements when the cut elements are already marked using the level-set method. Moreover, in Section 2.3.4, the details of embedding each cut element by a conforming subtriangulation for integration purpose is presented.

2.3.2. Refinement for invalid cut elements

In Section 2.3.1, we assume that elements of the background mesh are cut by the surface interface with *valid* cases: i.e. the intersection between a tetrahedron element and the surface is either a triangle or a quadrilateral, see Figure 9b. The number of intersection between the surface and the element edges is either 3 or 4. However, *invalid* cutting cases arise when, for example,

an edge of an element is cut by the surface with more than one intersection, or there are only two, or more than 4 edges of a tetrahedron, which are cut by the surface. These invalid cases can arrive when a coarse background mesh is used together with a *curved* interface surface. Figures 6a, 6b, and 6c schematically show, in two dimensions, some invalid cut cases between a triangle and an interface curve.

To overcome this issue, one solution, as in [39], is to *recursively* embed the invalid cut element with a set of sub-elements until all (sub) cut elements are valid, see Figures 6d, 6e, and 6f. In what follows, this procedure is called *refinement*. However, it is important to note that the invalid cut element is not actually refined since we do not introduce any new degree of freedom into the background mesh. The number of level of refinement needed to get all valid cut elements depends not only on the coarseness of the background mesh and curvature of the interface surface, but also on the relative location between the cut element and the surface. In Figures 6d, 6e, only one refinement level is needed in order to get valid cut elements, whereas in Figure 6f, two refinement levels are necessary. Once we get all valid cut elements, they are embedded by a conforming subtriangulation as usual for integration reason, described above. Figures 6g, 6h, and 6i show the conforming embedded elements for valid cut (sub) elements.

As an example to demonstrate the implemented algorithm which works on tetrahedra, Figure 7a shows a spherical surface which is immersed into a background mesh, and valid and invalid cut elements with the spherical surface are embedded with sub-tetrahedra shown in Figure 7c. Figure 7b shows four levels of refinement needed to capture the intersections between an invalid cut tetrahedron and the spherical surface.

2.3.3. Stabilisation by moving background nodes

There may be situations in which an element is cut by the surface with a very small intersection, as schematically shown in Figure 8a. These situations can strongly affect the stability of the system matrix. To remedy this stability

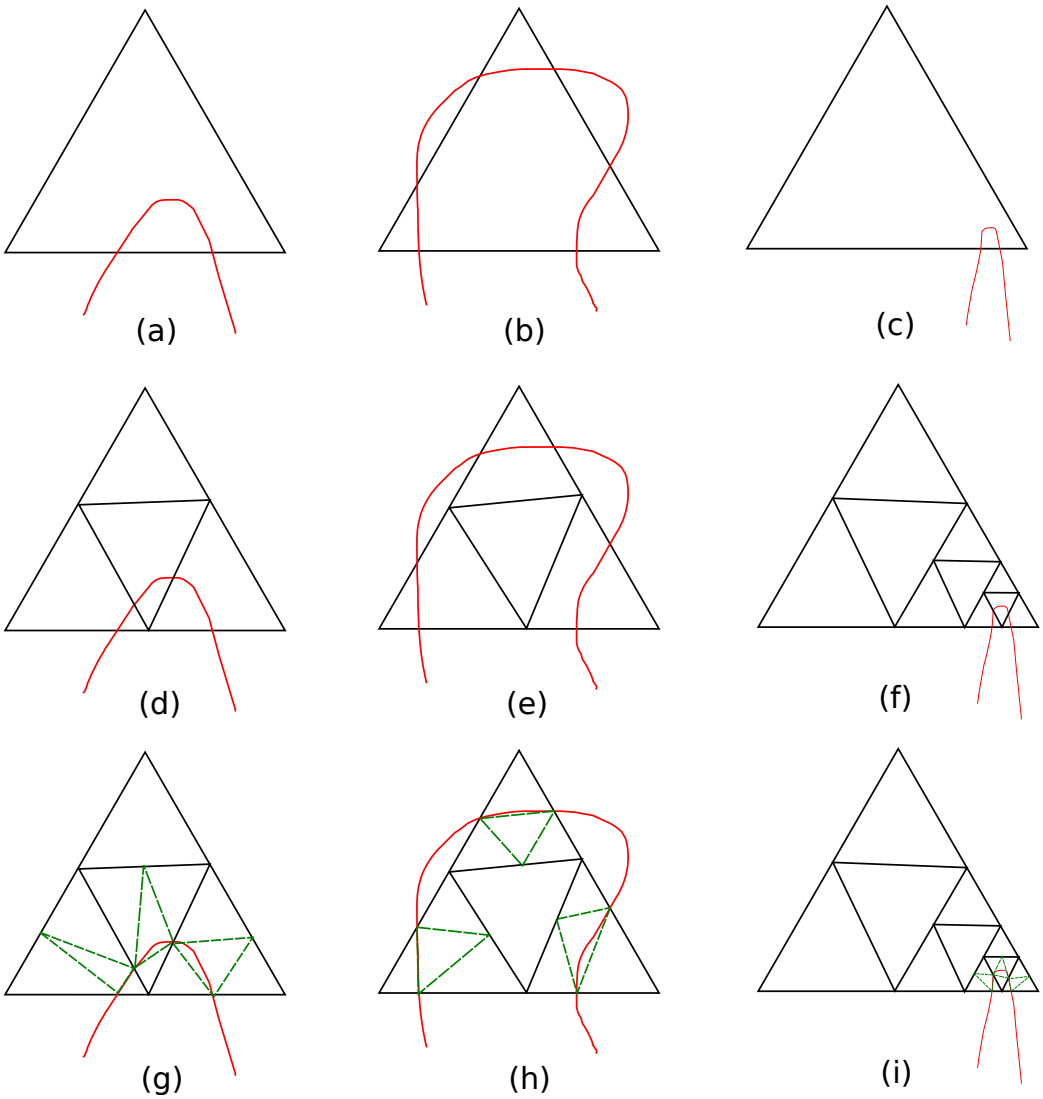
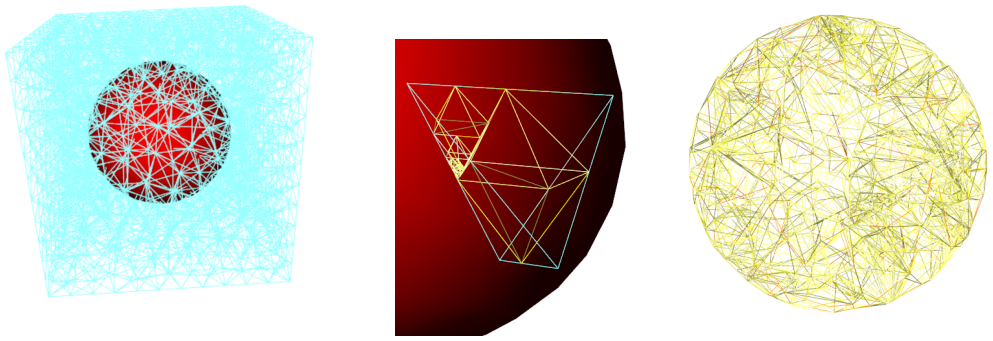


Figure 6: Schematic representation of invalid cut elements in two dimensions: (a), (b), (c); refinements lead to valid cut elements: (d), (e), (f). Valid cut elements are embedded with a subtetrahedron set (shown in green colour) for integration purpose: (g), (h), (i).

issue, we propose to move the concerned node by a random distance, which is proportional to the element size, defined by the vector

$$\mathbf{d} = -\frac{1}{2}\beta \cdot l_e \cdot \mathbf{n}, \tag{29}$$



(a) (b) (c)

Figure 7: A sphere is immersed into a background mesh (a), an invalid cut element is embedded with sub-tetrahedra after refinement (b), and finally all, invalid and valid, cut elements, intersected with the spherical surface, are embedded with sub-tetrahedra, with or without refinement, respectively, for integration reason (c).

with $0 < \beta < 1$ a random variable, l_e the element size, and $\mathbf{n}$ the outward unit normal of the surface. In practice, β is set between 0.1 and 0.9 which can efficiently overcome the issue. This approach is schematically shown in Figure 8b.

We define a criterion to detect when the intersection between an element and a surface is small, and thus the moving node approach is applied. This criterion reads

$$\frac{V_i}{V_e} < \alpha, \tag{30}$$

where V_i is the volume of the smaller part of the element cut by the surface, and V_e is the total volume of the element. In this study, we set α to 0.05, i.e., when the volume of the smaller part of the element is less than 5% of the total volume of the element, we move the node.

It is important to note that this moving node approach is also applicable when refinement method described in Section 2.3.2 fails to determine valid cut elements. In fact, if after five levels of refinement, invalid cut elements are still present, we simply move, by a distance vector expressed in Equation (29), the element node which is closest to the intersection between the element and

the surface.

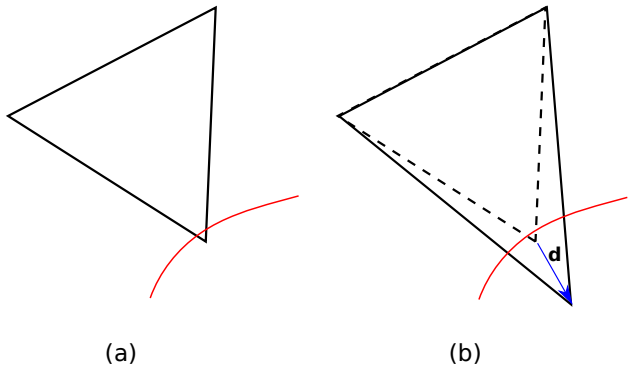


Figure 8: When an element is cut by the surface with a small intersection (a), the concerned node is moved by a distance proportional to the element size (b).

2.3.4. Implementation aspects

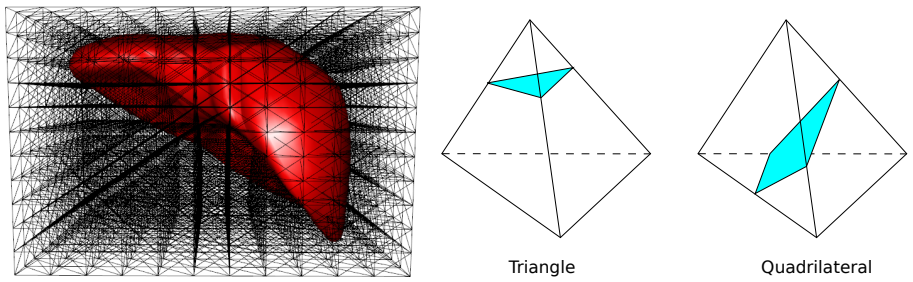
To discuss the implementation aspects, as a starting point, we assume that an arbitrary surface is immersed into a computational background mesh. An example of a liver surface being immersed in a computational tetrahedron mesh is shown in Figure 9a.

Since the interface surface is arbitrary, and may be concave, due to the change of orientation of the outward normal of the surface, it may not be efficient to use only level-set function to distinguish between three types of elements described above. Moreover, to mark those elements which are far from the surface, using the distance function as level-set function, in combination with outward normal of the surface, raises the ineffectiveness of the algorithm. To remedy this issue, we mark different types of elements using the following approach, see also Figure 10a.

Step 1 Using level-set function (as described in Section 2.3.1), mark the cut elements by checking the potential intersections between each triangle of the surface mesh with immediate neighboring elements (tetrahedra) of the triangle. Label these elements by 1,

Step 2 Mark the outside elements by propagating the marking procedure, starting from the elements located at the boundary $\partial\Omega$ of the domain until a cut element is reached. Label these elements by 0,

Step 3 Mark the remaining elements as inside. Label these elements by 2.



(a) (b)

Figure 9: A liver surface is immersed in a computational tetrahedron mesh (a), and intersection cases between a surface and a tetrahedron (b).

To identify which tetrahedra are located around each triangle of the surface, to check for potential intersections in Step 1, and to classify the tetrahedra located at the boundary domain $\partial\Omega$ (for marking procedure to propagate from) in Step 2, we use the following approach. We first compute the bounding box of the domain. The bounding box is then subdivided into subcubes, as shown in Figure 10a. All subcubes which are incident to each triangle of the surface, see Figure 10b, are then computed. All tetrahedra which are incident to each subcube are also figured out, see Figure 10c. From these two data structures, all cut elements can be marked because one can easily access the tetrahedra around each triangle on the surface to check for intersections. On the other hand, to propagate marking procedure for outside elements, all tetrahedra located at the boundary $\partial\Omega$ can also be easily accessed from the boundary subcubes.

Figure 11 shows three type of tetrahedra marked when a liver surface is immersed in a tetrahedron mesh.

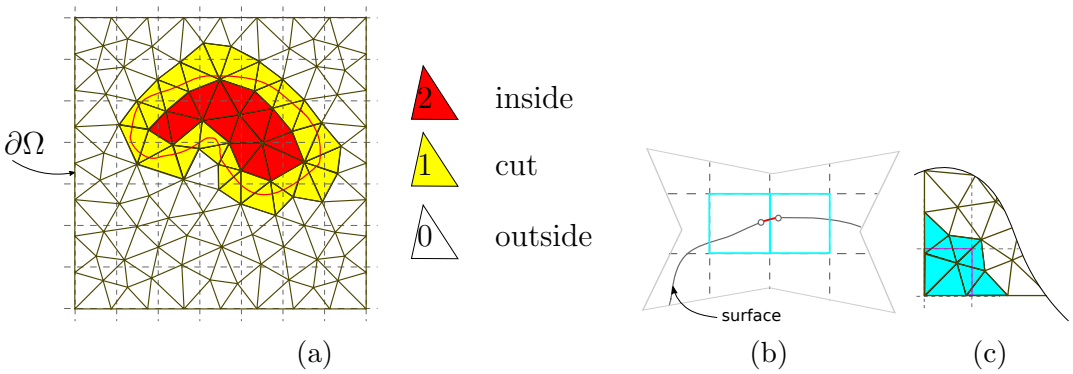


Figure 10: Subdivision of the bounding box (a), subcubes incident to each triangle of the interface are computed (b), elements incident to a subcube are marked (c).

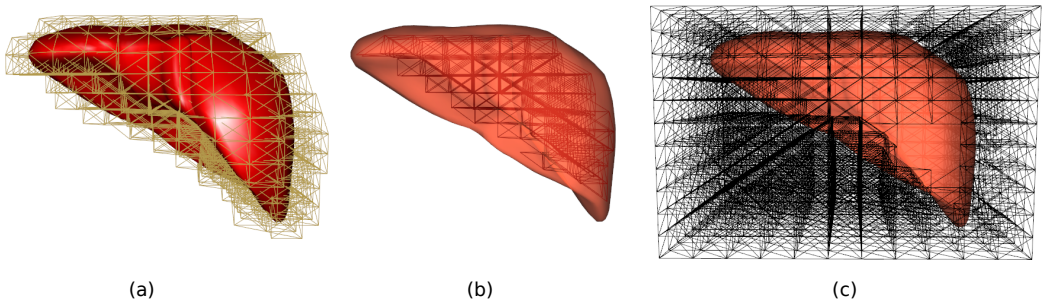


Figure 11: Three type of tetrahedra are marked when a liver surface is immersed in a computational tetrahedron mesh: cut element (a), inside elements (b) and outside elements (c).

Once all cut elements are identified, to facilitate the integration, a set of sub-tetrahedra with conforming nodes (regarding the interface surface) is embedded in each cut tetrahedron. There are only two kinds of intersection between a surface and a tetrahedron: a triangular intersection or a quadrilateral intersection, see Figure 9b. Therefore, it is sufficient to employ a set of eight tetrahedra to embed for each cut tetrahedron, see Figure 12a. This set is called the template set. Depending on each real case, where the tetrahedron is cut by a surface with a triangular intersection or a quadrilateral one, the template set is rotated and then mapped into the cut tetrahedron geometry using the mapped mesh method [40, 27]. It is noted that the template nodes

4, 5, 6, 7, 8, 9, see Figure 12a, are located at the middle of their corresponding edges. Having determined the real intersections between the cut tetrahedron and the interface, as described in Section 2.3.1, if any edge is intersected by the interface surface, the corresponding midpoint is moved to match the real intersection.

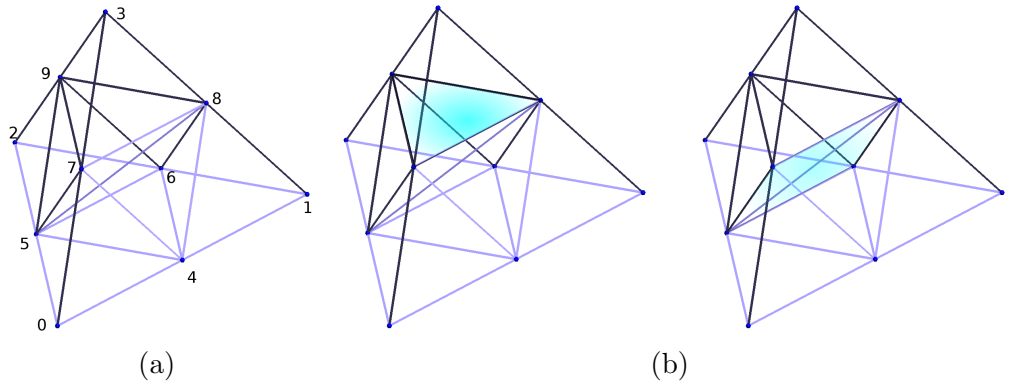
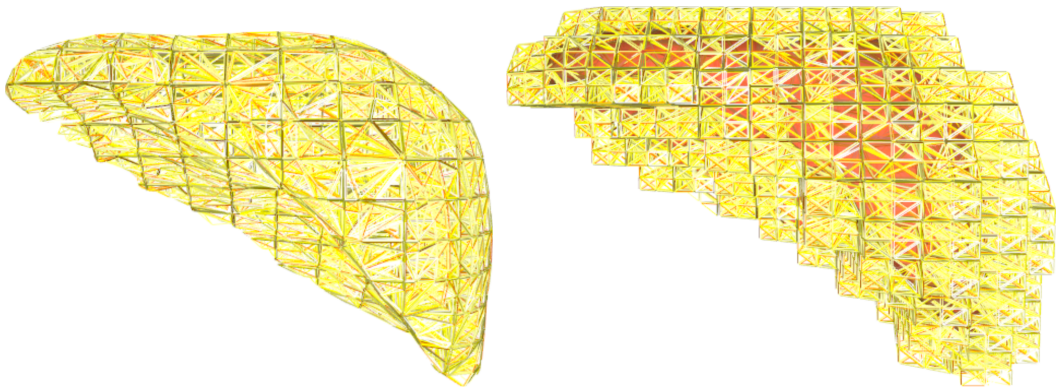


Figure 12: A set of eight template tetrahedra (a). Depending on whether the intersection between a surface and a cut tetrahedron is a triangular section or a quadrilateral one, the template is rotated and then mapped in the cut tetrahedron geometry (b).

Once a template set is embedded for each cut tetrahedron, the last step consists in computing the relative location (inside or outside) of the sub-tetrahedra of the template with respect to the interface surface. For this purpose, we again use the same level-set method which was used for computing the relative location (inside or outside) of the tetrahedra. Figure 13 shows the sub-tetrahedra which are marked inside and outside with respect to the liver surface.

As described in Section 2.3.2, when an invalid cut element arises, it is recursively embedded with a tetrahedron set from a predefined template. It results in a tree data structure as schematically shown in Figure 14. To efficiently handle this type of data structure in implementation, we use a STL-like C++ tree class `http://tree.phi-sci.com/`. Using this container, it allows to recursively add embedded elements as children of the cut element (regarded as parent) very easily. The container also provides different kinds



(a) (b)

Figure 13: Once a template tetrahedron set is embedded for each cut tetrahedron, using the level-set method, sub-tetrahedra are marked as inside (a) or outside (b) with respect to the interface surface.

of iterators to access desired elements efficiently.

2.3.5. Numerical integration

For those elements that are fully contained in the domain Ω_1 or Ω_2, see Figure 4a, integration of element stiffness and mass matrices are performed normally, as in classical FEM. Only for cut elements, the integration is split into two parts which are related to the inside and outside sub-tetrahedra with respect to the interface. A two dimensional schematic representation of the integration using natural coordinates on reference element is shown in Figure 15.

The stiffness matrix of the cut element reads

$$\mathbf{K}_e = \mathbf{K}_e^{\Omega_1} + \mathbf{K}_e^{\Omega_2}, \tag{31}$$

where $\mathbf{K}_e^{\Omega_1}$ and $\mathbf{K}_e^{\Omega_2}$ denote the stiffness contributions of the part belonging to Ω_1 and Ω_2, respectively, to the element e. The stiffness matrix on each part is computed by summing the contributions from their sub-elements. For

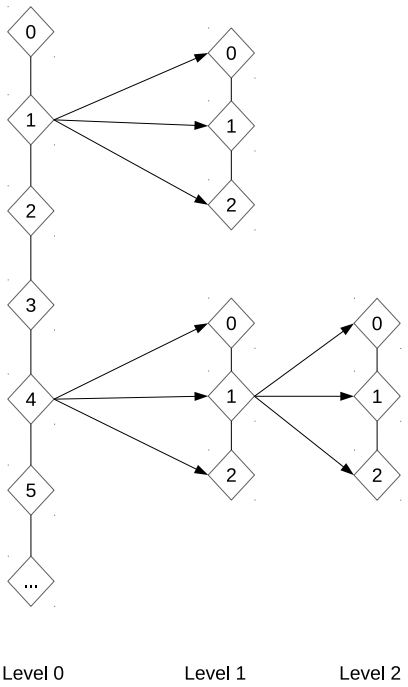


Figure 14: Schematic representation of a tree data structure obtained when cut elements are embedded with a subtetrahedron set or when an invalid cut element is recursively refined to get all valid cut elements before embedding with a subtetrahedron set. The elements of the background mesh are indexed at level 0, while embedded elements are indexed at subsequent levels.

example, the stiffness matrix $\mathbf{K}_e^{\Omega_2}$ reads

$$\mathbf{K}_e^{\Omega_2} = \sum_{s \in \Omega_2^k} \int_{\Omega_2^k} \mathbf{B}_s^T \mathbf{E}_2 \mathbf{B}_s \, \mathrm{d}\Omega_2^k = \sum_{s \in \Omega_2^k} \sum_{i=1}^{N_p} \mathbf{B}_s(\boldsymbol{\xi}_i)^T \mathbf{E}_2 \mathbf{B}_s(\boldsymbol{\xi}_i) \omega_i \det(\mathbf{J}(\boldsymbol{\xi}_i)), \quad (32)$$

in which $\mathbf{B}_s$ is the strain displacement matrix of the sub-element s, $\mathbf{E}_2$ is the material stiffness tensor of the domain Ω_2, $\boldsymbol{\xi}_i$ and ω_i are the quadrature coordinates and the corresponding weight parameters, N_p is the number of quadrature points used, and $\mathbf{J}$ is the Jacobian matrix of the coordinate transformation. Since the stiffness matrix should be expressed on the cut (parent) element where the degrees of freedom are defined, we must compute

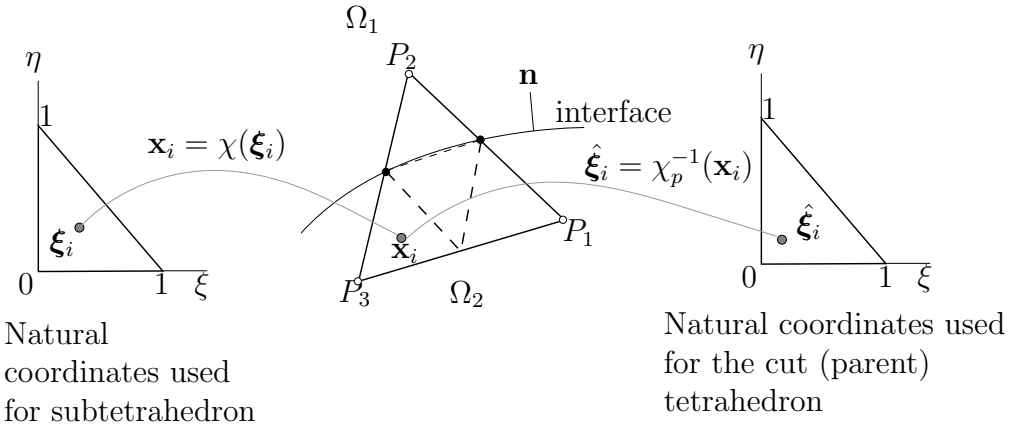


Figure 15: Integration on cut elements. Note that the degrees of freedom are only defined on the element nodes P_1, P_2 and P_3. χ and χ_p are the coordinate mapping from the reference element to the physical sub-element, and to the physical cut (parent) element, respectively.

the strain displacement matrix $\mathbf{B}_p$ of the parent element at the quadrature point $\hat{\boldsymbol{\xi}}_i$ corresponding to the physical coordinates $\mathbf{x}_i$ of the sub-element, see Figure 15. So, Equation (32) reads

$$\mathbf{K}_e^{\Omega_2} = \sum_{p \in \Omega_2^k} \sum_{i=1}^{N_p} \mathbf{B}_p(\hat{\boldsymbol{\xi}}_i)^T \mathbf{E}_2 \mathbf{B}_p(\hat{\boldsymbol{\xi}}_i) \omega_i \det(\mathbf{J}(\hat{\boldsymbol{\xi}}_i)), \tag{33}$$

where $\mathbf{B}_p$ is the strain displacement matrix defined on the parent element $P_1P_2P_3$.

In this paper, we use linear tetrahedra. Therefore, the strain displacement matrix is constant across the element volume, and $\omega_i = 1/6$, $\det(\mathbf{J}) = 6V_k$ with V_k the sub-element volume, see e.g. [41] for more details. Thereby, we get

$$\mathbf{K}_e^{\Omega_2} = \sum_{\Omega_2^k} \mathbf{B}_p^T \mathbf{E}_2 \mathbf{B}_p V_k. \tag{34}$$

The computation for $\mathbf{K}_e^{\Omega_1}$ can be done using the same concept. Also, integration over cut elements for the mass matrices is performed by the same

procedure.

2.4. Corotational formulation for CutFEM

In many surgical simulations, tissues undergo large displacements and rotations, see e.g. [42, 43]. Using linear elasticity for modelling of soft tissues results in artifacts for large rotational deformation [33]. To overcome this issue, we compute the stiffness matrix using the corotational formulation as in [32], in which the rigid motion can be extracted from the total finite element displacements.

Indeed, the system term $\boldsymbol{K}\boldsymbol{u}$ of Equation (17) is computed by assembling the element internal force, denoted by $\boldsymbol{f}_e^i$. In the following, the superscript i is dropped for simplicity. The element displacement vector $\boldsymbol{u}_e$ is computed from the rotated current configuration and the initial one as

$$\boldsymbol{u}_e = \boldsymbol{R}_e^T \mathbf{x}_e - \mathbf{x}_{\mathbf{0}e}, \tag{35}$$

where $\boldsymbol{R}_e$ stands for the element rotation matrix of the element local frame with respect to its initial orientation, being updated at each time step. The corotated element internal force is then computed as

$$\boldsymbol{f}_e = \boldsymbol{R}_e \boldsymbol{K}_e \boldsymbol{u}_e. \tag{36}$$

Using polar decomposition, the element rotation matrix $\boldsymbol{R}_e$ is computed from the element deformation gradient $\boldsymbol{F}_e$ as follows

$$\boldsymbol{R}_e \cdot \boldsymbol{U}_e = \boldsymbol{F}_e, \tag{37}$$

where $\boldsymbol{U}_e$ is the right stretch tensor that is responsible for deformation. The element deformation gradient $\boldsymbol{F}_e$ is computed as

$$\boldsymbol{F}_e = \boldsymbol{P}_n \cdot \boldsymbol{N}^{'}, \tag{38}$$

where $\boldsymbol{P}_n$ is the element nodal coordinates, and $\boldsymbol{N}^{'}$ is the derivative of shape functions. For tetrahedron element, $\boldsymbol{P}_n$ and $\boldsymbol{N}^{'}$ are 3×4 and 4×3 matrices, respectively

$$\boldsymbol{P}_n = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{pmatrix}, \tag{39}$$

$$\boldsymbol{N}^{'} = \begin{pmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial z} \\ \frac{\partial N_2}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial z} \\ \frac{\partial N_3}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial z} \\ \frac{\partial N_4}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial z} \end{pmatrix}. \tag{40}$$

The corotational formulation is summarised in Figure 16. For elements cut by the surface, since degrees of freedom are only defined on their nodes (not on nodes of subelements), see also Section 2.3.5, the element rotation matrix $\boldsymbol{R}_e$ is still computed by the procedure described above.

2.5. Boundary conditions on immersed surfaces

When using immersed surface, one needs to impose Neumann and Dirichlet boundary conditions on the immersed surface, which does not conform with the computational mesh. For the Neumann boundary condition, a force applied on the surface is barycentrically mapped onto the nodes of the element from the computational mesh which contains the applied point of that force, see Figure 17.

The concept of this approach is based on the master-slave scheme [44]. In this scheme, the displacement of a point on the surface can be seen as a mapping of the displacement of the computational mesh, which is given by

$$\mathbf{u}_S = \mathbf{J}\mathbf{u}_M, \tag{41}$$

where $\mathbf{u}_S$ is the displacement of the surface (considered as *slave*), $\mathbf{u}_M$ is the displacement of the computational mesh (regarded as *master*), and $\mathbf{J}$ denotes

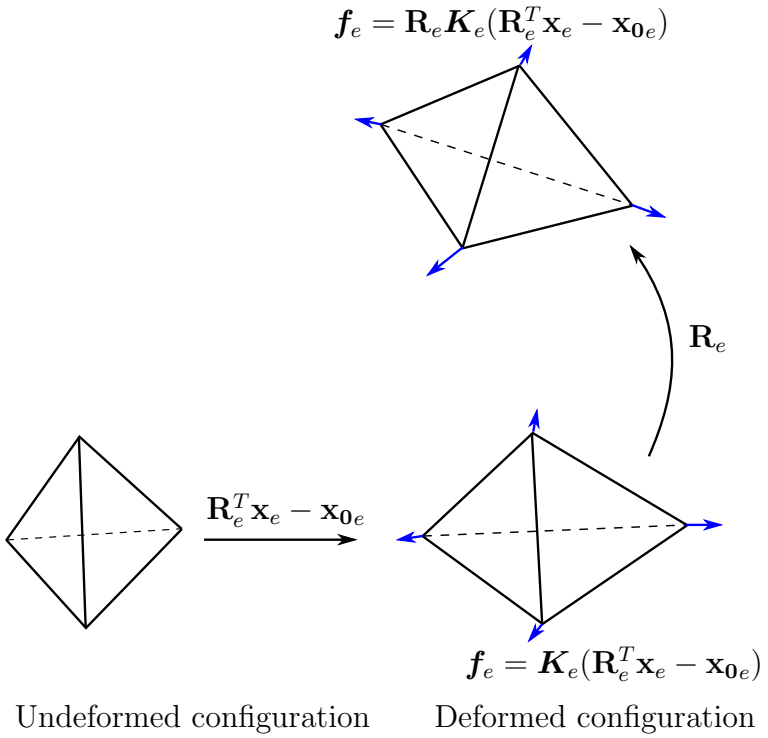


Figure 16: By using corotational formulation, the rigid body motion is removed from the element deformation.

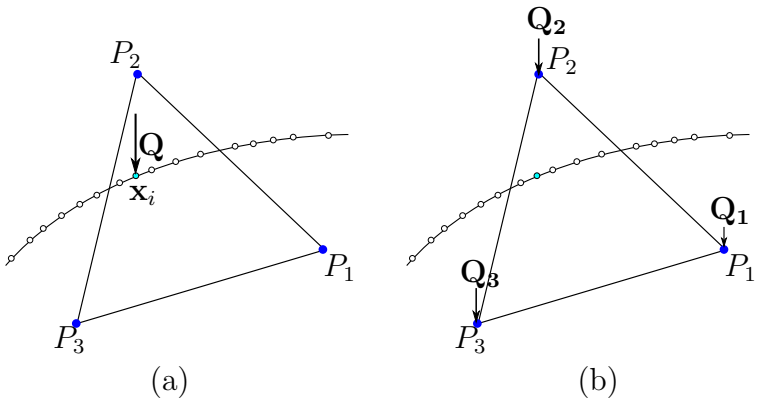


Figure 17: A force $\mathbf{Q}$ applied on the surface at point $\mathbf{x}_i$ (a), is barycentrically mapped into the nodes of the element from the computational mesh containing $\mathbf{x}_i$ (b).

the matrix containing barycentric coordinates (the relative coordinates) of the considered point on the surface with respect to the computational mesh.

For example, if the slave point is located at the centroid of the triangle, the relationship described by Equation (41) is explicitly expressed as

$$\begin{bmatrix} u_S \\ v_S \end{bmatrix} = \begin{pmatrix} 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \end{pmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix}, \tag{42}$$

where u and v are two components of the displacement in two dimensional space, the subscripts 1, 2, and 3 denote three nodes of the triangle.

By applying the principle of virtual work

$$\mathbf{u}_M^T \mathbf{Q}_M = \mathbf{u}_S^T \mathbf{Q}_S, \tag{43}$$

where $\mathbf{Q}_S$ and $\mathbf{Q}_M$ are the forces applied on the surface, and the equivalent forces applied at the computational mesh, respectively. By substituting Equation (41) into Equation (43), one obtains

$$\mathbf{Q}_M = \mathbf{J}^T \mathbf{Q}_S. \tag{44}$$

Dirichlet boundary conditions at some nodes i on the surface, can be imposed as $\mathbf{u}_S^i = \bar{\mathbf{u}}$, where $\bar{\mathbf{u}}$ is the prescribed displacement. Taking into account Equation (41), the Dirichlet boundary condition can be expressed as

$$\mathbf{J}\mathbf{u}_M = \bar{\mathbf{u}}. \tag{45}$$

Using Lagrange multipliers, one can easily impose the prescribed displacement

on the computational mesh by solving the system equation set

$$\begin{pmatrix} \mathbf{A} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix} \begin{Bmatrix} \boldsymbol{u}_M \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{b} \\ \bar{\mathbf{u}} \end{Bmatrix}, \tag{46}$$

where $\boldsymbol{\lambda}$ stands for Lagrange multipliers used for Dirichlet boundary conditions.

2.6. Solving system equations with constraints

To solve the system equations with constraints, there are several prominent approaches, each with their own advantages and disadvantages. The penalty method, where the respective matrix entries are multiplied by some large value, suffers from the uncertainty of the penalty parameter (weight). By using a very small value, the constraints may not be accurately imposed, and using a high value may lead to an ill-conditioned system (and thus directly affecting the computational time). The Nitsche's method is problem-dependent since it requires a penalization parameter for stability. Compared to these two approaches, the Lagrange multiplier method offers some advantages. It offers excellent generality and accuracy (no dependence on the problem), and it is almost insensitive to the user's choice (no parameter to choose). However, this method is no panacea either. Using Lagrange multiplier method, we have to solve for additional unknowns. Nevertheless, in our implementation for Lagrange multiplier method, we do not solve the system matrix with constraints by inverting the whole system (augmented with Lagrange multipliers) since it renders the stiffness matrix indefinite. Instead, we solve the constraints in three steps, as described in the following: i) solve the unconstrained system, ii) solve for constraints, and iii) update the solution when Lagrange multipliers are available. These steps are detailed in Algorithm 1. Note that, solving the system equations by using this approach is also relevant for parallelization, which is crucial for real-time simulations. We also remark that we do not employ any stabilization technique in the proposed Lagrange multiplier's

approach, such as that used in [11]. As stated in Section 2.2, for the tissue we employ C^0-nonconforming three-dimensional linear elements, and for the needle we utilise C^1-conforming one-dimensional linear beam elements (see the end of Section 2.2). The numerical results presented in Section 3, and those presented in [27, 28], show that the proposed Lagrange multiplier approach, together with this choice of spaces, delivers converging results, and that it does not suffer from any locking or instability issues. We intend to conduct a rigorous mathematical analysis of the proposed method in our future work.

The interaction between the needle (denoted by subscript 1) and the tissue (denoted by subscript 2) can be expressed by the following equation set

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{H}_1^T \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{H}_2^T \\ \mathbf{H}_1 & \mathbf{H}_2 & \mathbf{0} \end{pmatrix} \begin{Bmatrix} d\mathbf{v}_1 \\ d\mathbf{v}_2 \\ \boldsymbol{\lambda}_i \end{Bmatrix} = \begin{Bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \mathbf{0} \end{Bmatrix}, \tag{47}$$

where $\boldsymbol{\lambda}_i$ is the Lagrange multiplier representing the *interaction* between the needle and the tissue.

We can see that, Equation (47) describing the interaction between the needle and the tissue, and Equation (46) expressing the constraints used for Dirichlet condition on implicit boundaries, have the same general form

$$\begin{pmatrix} \mathbf{A} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix} \begin{Bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{b} \\ \mathbf{c} \end{Bmatrix}. \tag{48}$$

Equation (48) can be reformulated as

$$\mathbf{x} = \underbrace{\mathbf{A}^{-1}\mathbf{b}}_{\mathbf{x}_{free}} - \underbrace{\mathbf{A}^{-1}\mathbf{J}^T\boldsymbol{\lambda}}_{\mathbf{x}_{cor}}, \tag{49a}$$

$$\underbrace{\mathbf{J}\mathbf{A}^{-1}\mathbf{J}^T}_{\mathbf{W}}\boldsymbol{\lambda} = \mathbf{J}\underbrace{\mathbf{A}^{-1}\mathbf{b}}_{\mathbf{x}_{free}} - \mathbf{c}, \tag{49b}$$

in which, $\mathbf{x}_{free}$ can be seen as the solution of the unconstrained system

$\mathbf{Ax} = \mathbf{b}$, $\mathbf{x}_{cor}$ is the corrective solution due to constraints, which can be seen as the solution of the system $\mathbf{Ax}_{cor} = \mathbf{J}^T\boldsymbol{\lambda}$. Here $\mathbf{W}$ denotes the compliance matrix. As described in [28], to be able to solve the Equation (48) in real time, we use the advanced method described in Algorithm 1.

Algorithm 1 Algorithm for solving the system equation with constraints (Equation (48)). $\mathbf{L}$ denotes a lower unit triangular matrix, and $\mathbf{D}$ is a diagonal matrix.

1: $\mathbf{LDL}^T \leftarrow decompose(\mathbf{A})$ ▷ (Cholesky decomposition)
2: $\mathbf{x}_{free} \leftarrow solveCholesky(\mathbf{Ax}_{free} = \mathbf{b})$ ▷ (Solve unconstrained problem)
3: $\boldsymbol{\lambda} \leftarrow solve(\mathbf{W}\boldsymbol{\lambda} = \mathbf{Jx}_{free} - \mathbf{c})$ ▷ (Solve for constraints, see (49b))
4: $\mathbf{x}_{cor} \leftarrow solveCholesky(\mathbf{Ax}_{cor} = \mathbf{J}^T\boldsymbol{\lambda})$ ▷ (Compute corrective solution)
5: $\mathbf{x} \leftarrow \mathbf{x}_{free} - \mathbf{x}_{cor}$ ▷ (Update solution, see (49a))

Therefore, three main steps for solving Equation (48) can be summarised as

Step 1: Decompose the matrix $\mathbf{A}$, and solve for $\mathbf{x}_{free}$, described by Lines 1 and 2 of Algorithm 1,

Step 2: Use $\mathbf{x}_{free}$, solve for Lagrange multipliers $\boldsymbol{\lambda}$ described in Line 3,

Step 3: Solve for the corrective solution $\mathbf{x}_{cor}$, and update the solution $\mathbf{x}$, described in Lines 4 and 5.

It is noted that the compliance matrix $\mathbf{W} = \mathbf{JA}^{-1}\mathbf{J}^T$ is computed from the decomposition $\mathbf{LDL}^T$ of the matrix $\mathbf{A}$, and this computation is parallelized on GPU, as proposed in [6, 28].

3. Results

In this section, we present numerical results with the following goals:

1. To measure the convergence of the proposed implicit boundary method as compared to its conforming counterpart (classical FEM) in terms of

the ability to reproduce the geometry of the organ/domain. This is the focus of Section 3.1, and partly of Section 3.2.

2. To measure the ability of the implicit boundary method in dealing with boundary conditions which are not imposed upon the mesh, but upon an implicitly defined boundary. This is the focus of Section 3.2-Section 3.4.

3.1. Convergence study

Before proceeding to study the convergence of the numerical solution obtained using CutFEM, we first study how accurately the non-polygonal geometry is approximated in CutFEM. In order to measure the ability of the method to represent the exact geometry, we compute the volume of the geometry discretised by CutFEM. Since the input of the geometry of the organ, e.g., liver or brain, is given in a discretised form, we compare the volume obtained using CutFEM against the volume obtained from the discretised surface. We consider the discretised surface of a spherical geometry with the diameter $d = 1.4$. As shown in Figure 18, the volume error, between the volume obtained from the discretised surface ($V_e = 1.40005$) and the volume V of the sphere discretised by CutFEM (based on the embedded discretised surface), converges with an optimal rate, which is approximatively the same rate as the L_2 norm of the displacement error.

In order to verify the reliability of the solution obtained using CutFEM, now we study the convergence of the solution under mesh refinement. We consider two cases, a tensile test and a bending test. As shown in Figure 19, a beam is studied with a spherical surface being embedded inside. As mentioned above, the spherical surface, with the diameter $d = 1.4$, is given in the discretised form. At one end, the beam is subjected to a uniform horizontal pressure (tensile) or a uniform vertical pressure (bending), while the other end is clamped. For the convergence study, the linear elastic constitutive law is used. The mechanical properties, namely, the Young's modulus and the Poisson's ratio, are denoted by E and ν, respectively. For the material outside

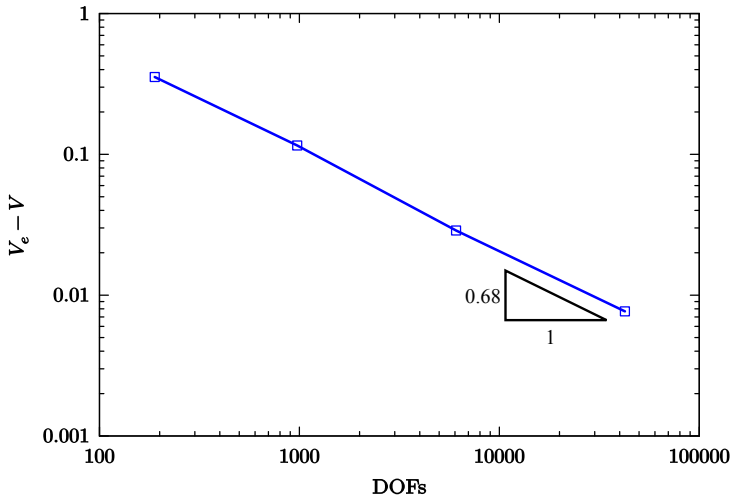


Figure 18: Convergence of the volume of the sphere, discretised by CutFEM from the embedded discretised surface, as compared to the exact volume obtained from the discretised surface ($V_e = 1.40005$), under background mesh refinement.

the sphere we denote them by E_1, ν_1, and for the material inside the sphere we denote them by E_2 and ν_2. The dimension of the beam is $6 \times 2 \times 2$ mm. In order to compare the convergences rate of the tests with the theoretical ones, we use the same mechanical properties for the material inside and outside the sphere surface. We set thus $E_1 = E_2 = 1000$ MPa, and $\nu_1 = \nu_2 = 0.1$.

The convergence is studied by computing the solution of the tensile and bending tests employing the tetrahedral meshes consisting of $7 \times 3 \times 3$, $13 \times 5 \times 5$, $25 \times 9 \times 9$, and $49 \times 17 \times 17$ nodes. We propose to use the solution from the classical FEM when employing a very fine mesh ($97 \times 33 \times 33$ nodes) as the reference solution. We then study the convergence of the error between the CutFEM solution and the reference FEM solution. This error is measured by using both, the L_2 norm and the energy norm. The L_2 norm of the displacement error reads

$$\|\eta\|_{L_2} = \sqrt{\frac{\int_\Omega (u_h - u_r)^2 \, \mathrm{d}\Omega}{\int_\Omega u_r^2 \, \mathrm{d}\Omega}}, \tag{50}$$

where u_h denotes the displacement solution of the CutFEM, and u_r denotes the reference solution of the classical FEM. The energy norm is defined as

$$\|\eta\|_{Energy} = \sqrt{\frac{\int_\Omega (\sigma_h - \sigma_r) \cdot (\epsilon_h - \epsilon_r)\, \mathrm{d}\Omega}{\int_\Omega \sigma_r \cdot \epsilon_r \, \mathrm{d}\Omega}}, \tag{51}$$

where σ_h and ϵ_h denote the stress and strain of the CutFEM, respectively, and σ_r and ϵ_r denote their reference values obtained from the classical FEM on a very fine mesh.

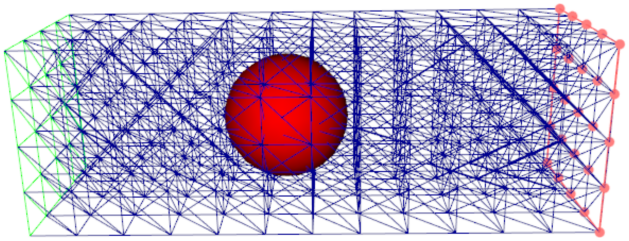


Figure 19: A beam is clamped at the right end (marked by the red points), and is subjected to, at the left end (marked by the green triangles), a uniform horizontal pressure (the tensile case) or a uniform vertical pressure (for bending case). A sphere surface is immersed inside the beam geometry.

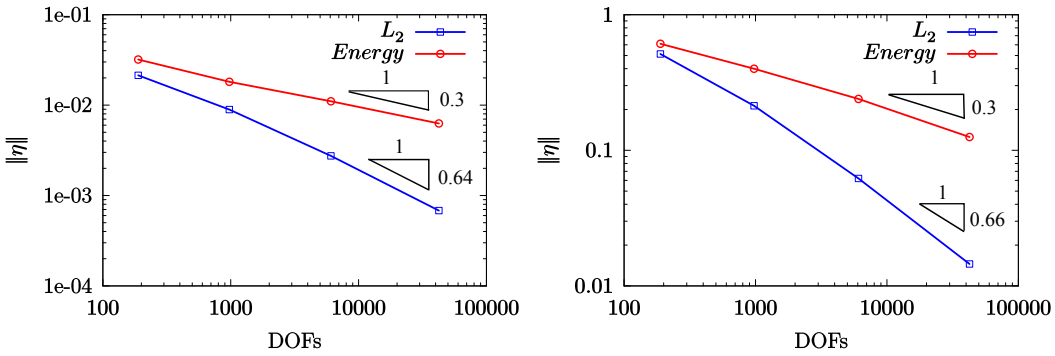


Figure 20: Convergence rates under mesh refinement for the tensile test (a), and for the bending test (b).

For tensile and bending tests, Figure 20 shows the convergence rates in

the L_2 norm and the energy norm versus the number of DOFs. It is observed that for both tests, the rates of convergence of the L_2 norm and the energy norm agree well with the theoretical rates. Indeed, for 3D problems using linear elements, the L_2 norm of the displacement error is of order $O(N^{-2/3})$, while the energy norm converges with an order of $O(N^{-1/3})$, where N denotes the total number of the degrees of freedom.

3.2. A comparison with FEM

The aim is to compare displacement simulation results obtained from the CutFEM using non-conforming meshes with those obtained from the classical FEM using conforming meshes. We carry out the study on two different geometries: the simple beam geometry with an immersed sphere as shown in Figure 19, and the more complex liver geometry as shown in Figure 9a.

For the beam geometry, the dimensions and the mechanical properties are same as those used in Section 3.1. The displacement measured at the centre of the left end of the beam is employed to compare between the CutFEM (where the sphere is modelled implicitly) and the classical FEM. The beam is subjected to a uniformly distributed pressure $q = -5$ N/mm^2 in the vertical direction at the left end, whereas it is clamped at the right end. Under various mesh refinements, Figure 21 shows the displacement of the point during simulations using the CutFEM and the classical FEM. It is observed that, with the same number of degrees of freedom used, the results obtained from the CutFEM perfectly agree with that of the classical FEM. Also, under mesh refinement, the displacement asymptotically convergences to the solution of the fine mesh.

For the liver, due to its complex geometry, in order to apply the same boundary conditions acting on different conforming meshes (used for FEM), and on different non-conforming ones (used for the CutFEM), a homogeneous Dirichlet boundary condition is implicitly applied to the model through the points located on an imaginary cutting section, shown by the points in blue colour in Figure 22, whereas a uniformly distributed pressure $q = -1$ N/mm^2

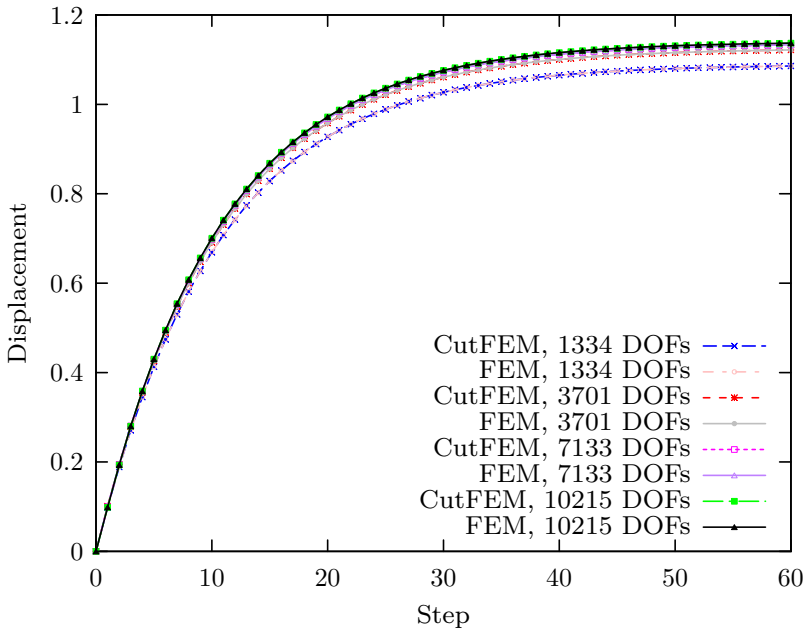


Figure 21: Displacement (in mm) measured at the centre of the left end of the beam during simulations by the CutFEM and the classical FEM.

is implicitly applied to the model in the vertical direction through the mesh shown by green colour in Figure 22. The displacement is measured at the point inside the liver shown by the grey colour in Figure 22. For these simulations, Young's modulus and Poisson's ratio are 1000, and 0.4 respectively.

Figure 23 shows the displacement of the point shown in Figure 22. It is observed that, at the same mean size of the elements used, the results obtained from the CutFEM agrees well with those of the classical FEM. It is noted that the number of degrees of freedom is not used as the same input during the comparison between the CutFEM and the classical FEM since, with respect to the liver geometry, it does not characterise the same mesh resolution between conforming mesh (used for FEM) and the non-conforming mesh (used for CutFEM). Instead, the mean element size l is employed (see Figure 23).

We now briefly discuss the computational cost of CutFEM vs FEM. We

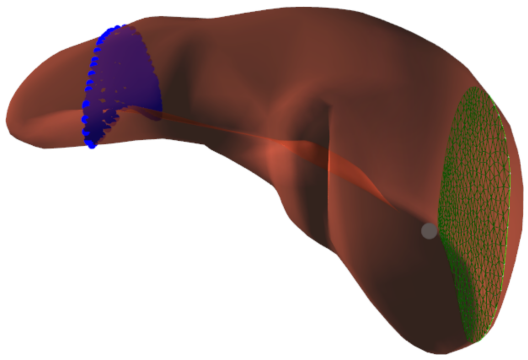


Figure 22: Liver is implicitly clamped at the points shown in blue colour, and is implicitly subjected to a uniformly distributed vertical pressure acting on the mesh shown in green colour. During the simulations, the displacement is measured at the point shown by the grey colour.

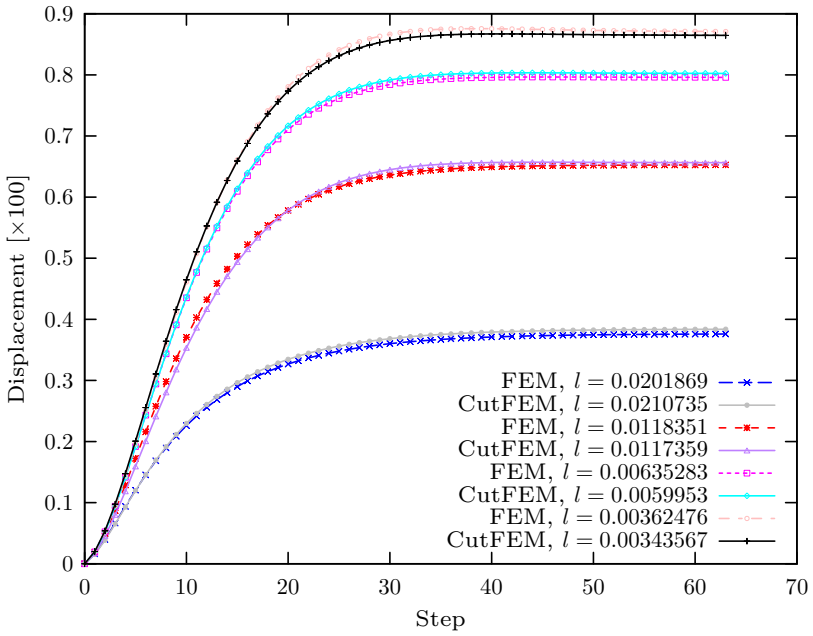


Figure 23: Displacement measured (in cm) at the point shown in Figure 22 during simulations by the CutFEM, and by FEM. Different mesh resolutions are denoted by different values of l.

consider the liver model for this comparison. For the simulation performed by FEM using a mesh with the mean mesh size $l = 0.00635283$, compared to

that performed by CutFEM using a mesh with $l = 0.0059953$, we observe a very good agreement on the displacement, see Figure 23. However, the use of CutFEM is more advantageous than the use of FEM since the computational times needed to solve the system equation by using FEM and CutFEM are 983.41 ms and 429.85 ms, respectively. Thus, for this mesh size, CutFEM is faster than FEM by a factor of approximately 2.3.

3.3. Needle insertion simulations

Having established the accuracy and reliability of CutFEM simulations as compared to the classical FEM, we now employ the CutFEM approach for needle insertion problems.

3.3.1. Immersed interface

The needle, which is initially inclining at an angle of 3.5 degrees, is inserted into a phantom tissue with a simple geometry, as shown in Figure 24. We simulate a spherical inclusion (can be considered as a tumor) which is implicitly immersed in the tissue phantom. The dimension of the tissue phantom is $6 \times 2 \times 2$, while the radius of the inclusion is 0.7. The length of the needle is 2.8, and its cross section radius is 0.05. Young's modulus and Poisson's ratio of the needle is set to 20 000 and 0.2, respectively. These parameters for the phantom tissue are $E_1 = 1\,000$ and $\nu = 0.4$. The Poisson's ratio of the inclusion is also set to 0.4, however, in order to investigate the effect of the inclusion stiffness on the needle-tissue interaction force profile, the Young's modulus of the inclusion E_2 is varied with respect to that of the phantom tissue by a factor of 1, 2, 4 and 8. The penetration strength at the tissue surface is set to 1, and the frictional coefficient between the tissue and the needle shaft is set to 0.5.

Figure 25 shows the needle-tissue interaction force with respect to the displacement of the needle tip, with different ratios E_2/E_1. It is observed that when the needle tip reaches the tissue surface, the interaction force between the needle and the tissue occurs. This interaction force continuously increases and

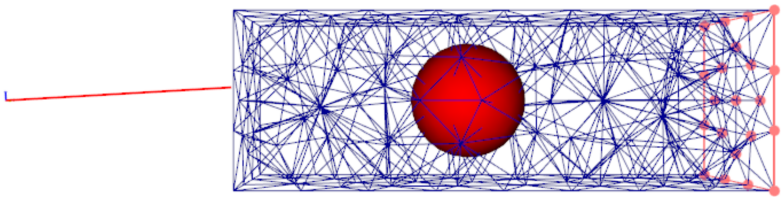


Figure 24: Schematic illustration of a needle insertion simulation into a simple tissue geometry. The needle is initially inclining at an angle of 3.5 degrees.

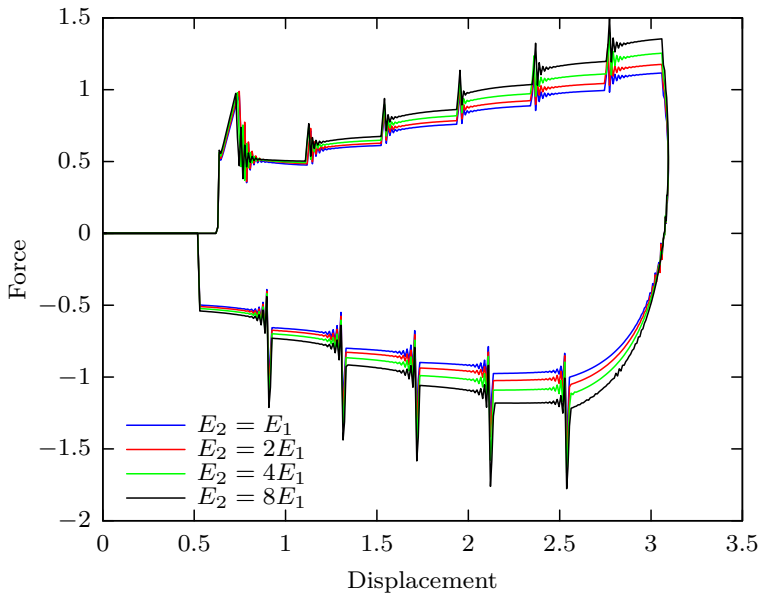


Figure 25: Force displacement curves, with varying Young's modulus ratio E_2/E_1, when the needle is inserted into the soft tissue model. When the needle tip displacement reaches about 3, the needle is retracted, and thus generating negative needle-tissue interaction force; at the stage when the needle is completely retracted from the tissue, the interaction force vanishes.

when it reaches the tissue surface penetration strength, the needle penetrates into the tissue. It also reveals that the closer to the inclusion the needle tip is, the more different the interaction force profiles are obtained when the ratio E_2/E_1 is varied. This is indeed logical due to the higher stiffness of the inclusion as compared to that of the tissue. When the displacement of the needle tip reaches 3, the needle is continuously retracted until completely out

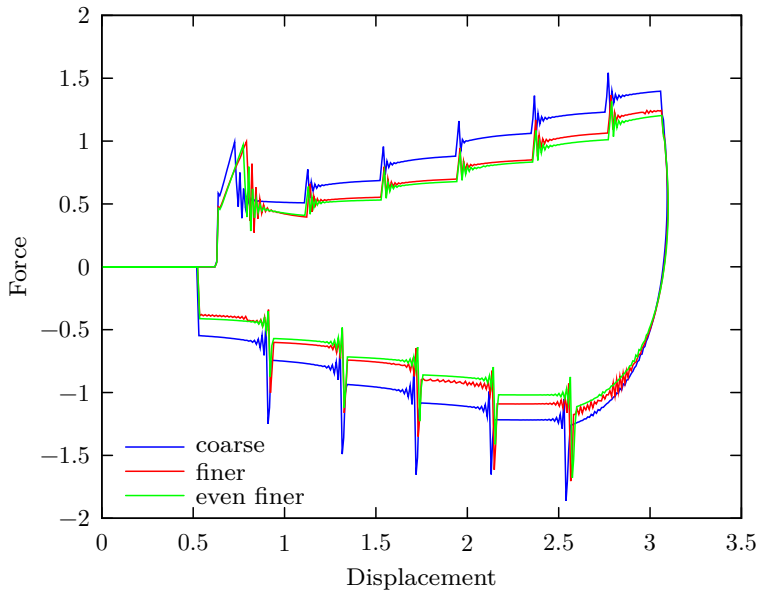


Figure 26: Force displacement curves, with varying mesh size (coarse mesh: 498 DOFs, finer mesh: 1575 DOFs, even finer mesh: 3183 DOFs) but for fixed Young's modulus ratio $E_2/E_1 = 8$, when the needle is inserted into the soft tissue model.

of the tissue. During retraction process, the interaction force changes its sign, and is negative, as observed in Figure 25. The same conclusion about the effect of inclusion-tissue stiffness ratio on the interaction force profile can be drawn during the retraction phase as during the insertion stage.

In Figure 26, we present the results of mesh refinement study during needle-tissue interaction, while keeping the ratio $E_2/E_1 = 8$ fixed. The curve profiles are qualitatively the same as of those curves presented in Figure 25. Moreover, it shows that when the mesh is refined, the proposed Lagrange multiplier method delivers convergent results (the difference between the red and green curves is smaller than that between the blue and red curves).

3.3.2. Fictitious boundary

We now consider a comparison between a needle insertion simulation into a liver using implicit/fictitious boundaries (CutFEM), and with those using

explicit/conforming boundaries (classical FEM). The goal is to show the performance of the CutFEM approach in dealing with Dirichlet boundary conditions defined implicitly on the surface, and in dealing with the interaction between the needle tip and the implicit surface (a kind of Neumann boundary conditions), as compared to those dealt explicitly by FEM. For the liver, Young's modulus and Poisson's ratio are set as 1 200 kPa and 0.4, respectively. For the needle, the same parameters as above are used. The background mesh used for simulation with CutFEM, and the boundary conditions applied to the liver surface, are shown in Figure 27. The point, where the displacement is measured during the insertion and the retraction of the needle, is also shown in Figure 27.

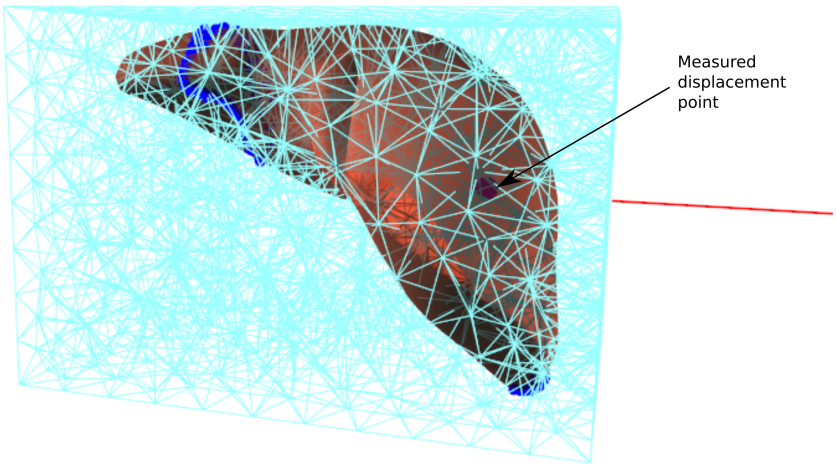


Figure 27: Background mesh used for the simulation of the liver behaviour during needle insertion and retraction. Some constraints, implicitly defined on the liver surface using Lagrange multipliers, are shown by the blue points. During the simulation, the displacement is measured at a point located near the needle shaft.

As can be seen in Figure 28, the displacement results obtained by CutFEM agree well with those obtained by classical FEM.

3.4. Electrode implantation simulation in Deep Brain Stimulation

The CutFEM is now employed to simulate an electrode lead implantation, using in Deep Brain Stimulation (DBS) procedure. We also take into account

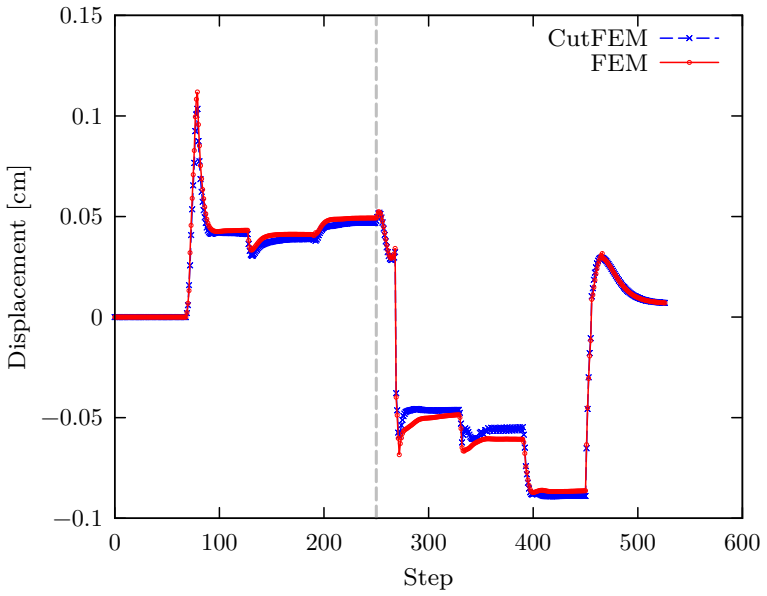
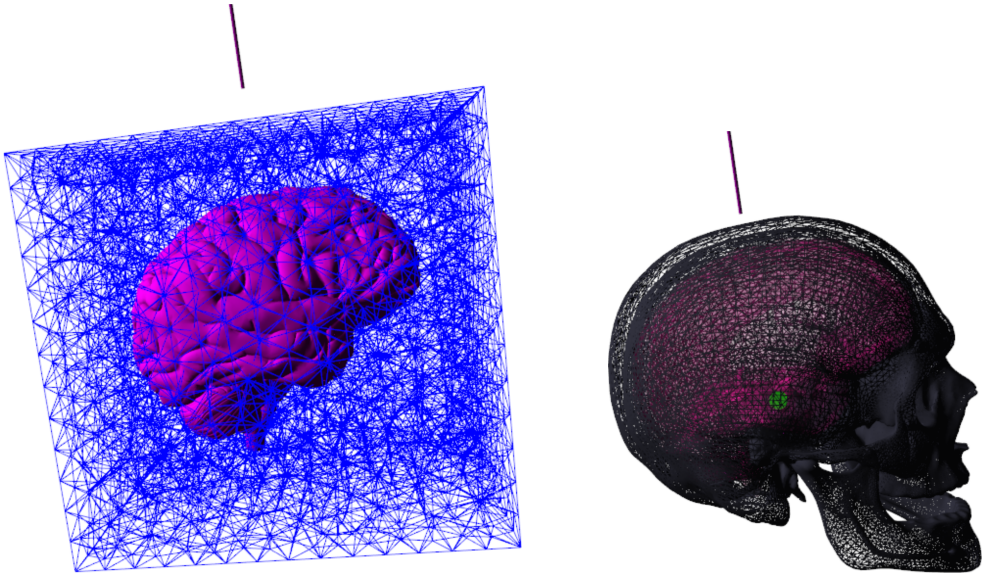


Figure 28: Displacement measured at the point during insertion and retraction, using CutFEM and standard FEM. The vertical dash line indicates the moment when the needle is starting to be retracted.

the brain shift phenomenon, which occurs due to the leak of cerebro-spinal fluid when a craniotomy is performed. The goal of the simulation is to insert an electrode inside the brain until it reaches the target, e.g., subthalamic nucleus (STN) area for treatment of Parkinson's disease. To do that, a cannula is inserted together with the electrode lead through a hole drilled in the skull. When they reach the STN area, the cannula is retracted while keeping the electrode lead inside. As in [43], frictional interactions between the brain tissue with the cannula and electrode lead are simulated. Young's modulus of 6 kPa and Poisson's ratio of 0.45 are set to the brain tissue. The cannula and electrode lead are set with Young's modulus of 10 GPa, and with Poisson's ratio of 0.3.

The input background mesh used for the CutFEM simulation of brain behaviour is shown in Figure 29a. We consider simple boundary conditions for the brain tissue. Indeed, brain tissue around the optic nerves and the

brainstem are considered to be clamped. Moreover, bilateral interaction constraints are considered between the brain surface and the skull. It is noted that these constraints are implicitly applied on the brain surface, as described in Section 2.5. During simulation, the displacement is measured at the point in the STN area as shown in Figure 29b.



(a) (b)

Figure 29: Input background mesh used for simulation of brain behaviour (a); during simulation, the brain has interaction with the skull, and the displacement is measured at the point shown in green colour (b).

Figure 30 shows the brain deformation at different stages of the simulation. The horizontal lines in Figure 30 help to show the differences of the brain deformation due to brain shift, and due to cannula insertion. The displacement of the STN target due to brain shift, and due to cannula insertion and full retraction, is presented in Figure 31. It reveals that the brain shift is the origin of the STN displacement which is stabilised about 1.1 cm. When the cannula is inserted inside the brain tissue, due to frictional interaction between them, the displacement of the STN target increases. The cannula is undergoing retraction immediately after the cannula tip has reached the STN

target. This induces the decreasing of the STN displacement before the STN target is stabilised around the location, where the STN was found after the brain shift stage.

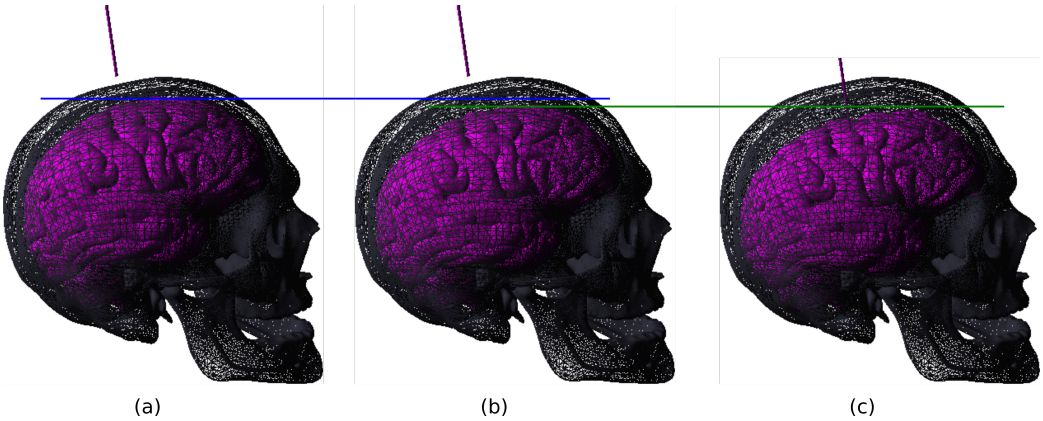


Figure 30: Brain at initial state (a), brain deformation due to brain shift (b), brain deformation due to cannula insertion (c). The blue horizontal line helps to visualise the brain deformation due to brain shift occurring after a craniotomy, whereas the green line helps to visualise the brain deformation due to the cannula insertion.

These results show that the CutFEM, even with a nonconforming mesh, is able to simulate the behaviour of the brain while it is in interactions with the skull, the cannula, and the electrode lead. This makes the discretisation as independent as possible from the geometric description. We believe that with such a tool, the patient-specific simulations (in terms of geometries, for instance) can be performed, particularly in the context of real-time simulations.

4. Conclusions

A corotational formulation of the CutFEM has been proposed. Using Lagrange multipliers, we have shown the methods to implicitly apply Dirichlet boundary conditions on the immersed surface, which is not conformed to the background mesh used for the simulations. We verified the implementation by studying convergences through a tensile test and a bending one. We also

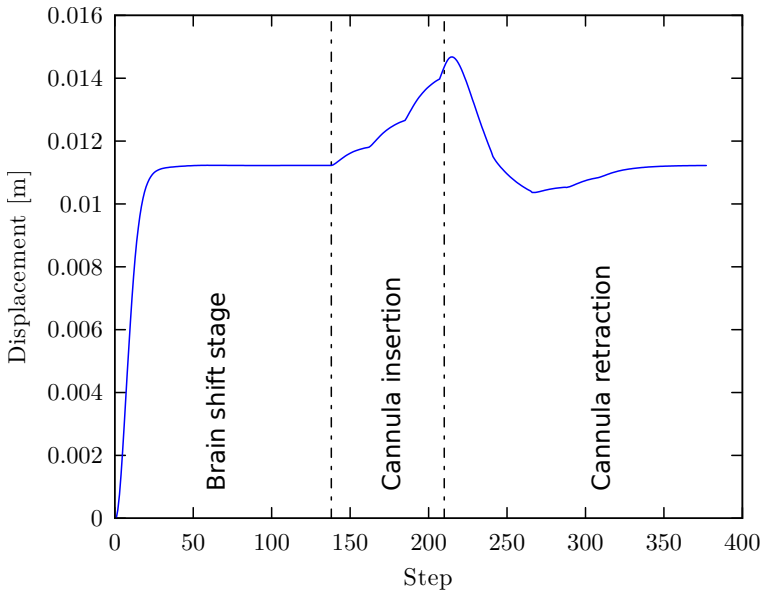


Figure 31: Displacement of the STN target due to brain shift, and due to cannula insertion and retraction.

demonstrated the performance of the CutFEM, compared to the classical FEM, through needle insertion problems in medical simulations (e.g. for biopsy or brachytherapy). We have shown that the interaction between the tissue surface being implicitly defined, and the needle tip, to account for the penetration phenomenon of the latter to the former (a kind of Neumann boundary conditions) during simulations, is working properly, and refers to the real-life situation. Also, by employing the CutFEM, it makes the discretisation as independent as possible from the geometric description, and also minimises the complexity of mesh generation, especially for complex geometries. Because of the constraint of computational time in real-time simulation context, while still preserving the geometric details, the use of coarse meshes becomes possible by using CutFEM.

Two kinds of applications using the CutFEM have been studied in this paper: i) immersed interfaces, and ii) fictitious boundaries. The immersed interfaces are useful for simulations of heterogeneity of tissues, e.g., when

tumors or internal structures of tissues are considered. Simulations using fictitious boundaries are suitable for applications in which the tissue geometries are complex, and by using the CutFEM, it is possible to integrate only the material inside the tissue surface while using a nonconforming mesh to its boundaries.

In the presented work, for the discretisation of simulated domains, we limited ourselves to linear elements. Another limitation is the embedding of linear subelements for the cut elements to facilitate numerical integration. By doing this, we still cannot precisely capture the surface geometries. For more accurate integration of implicit geometries, one can use higher order elements, as proposed in [45, 39]. We note that the isogeometric analysis, in the spirit of integrating finite element analysis and CAD design tools by using higher-order spline/NURBS basis functions from CAD geometry, could also be used for surgical simulations. We intend to study this approach in our future work. The interested readers are referred to [46] for pioneering work on isogeometric analysis, and a very recent work [47] on recovery-based error estimation and adaptivity over hierarchical T-meshes.

Acknowledgements

Stéphane Bordas, Satyendra Tomar and Huu Phuoc Bui thank partial funding for their time provided by the European Research Council Starting Independent Research Grant (ERC Stg grant agreement No. 279578) RealTCut "Towards real time multiscale simulation of cutting in non-linear materials with applications to surgical simulation and computer guided surgery". We are also grateful for the funding from the Luxembourg National Research Fund (INTER/FWO/15/10318764).

The funding from the University of Strasbourg Institute for Advanced Study (BPC 14/Arc 10138) for the first author is gratefully acknowledged.

Huu Phuoc Bui would particularly like to thank Dr. Davide Baroli, Dr. Lionel Untereiner, Dr. Paul Hauseux, Prof. Thomas-Peter Fries, Dr. Franz

Chouly, Prof. Alexei Lozinski, and Prof. Stéphane Cotin for many helpful discussions.

References

[1] S. R. Musse, D. Thalmann, Hierarchical model for real time simulation of virtual human crowds, IEEE Transactions on Visualization and Computer Graphics 7 (2) (2001) 152–164. doi:10.1109/2945.928167.

[2] M. Müller, L. McMillan, J. Dorsey, R. Jagnow, Real-Time Simulation of Deformation and Fracture of Stiff Materials, Springer Vienna, Vienna, 2001, pp. 113–124. doi:10.1007/978-3-7091-6240-8_11.

[3] S. Cotin, H. Delingette, N. Ayache, Real-time elastic deformations of soft tissues for surgery simulation, IEEE Transactions on Visualization and Computer Graphics 5 (1) (1999) 62–73. doi:10.1109/2945.764872.

[4] S. Cotin, H. Delingette, N. Ayache, A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation, The Visual Computer 16 (8) (2000) 437–452. doi:10.1007/PL00007215.

[5] C. Monserrat, U. Meier, M. Alcaiz, F. Chinesta, M. Juan, A new approach for the real-time simulation of tissue deformations in surgery simulation, Computer Methods and Programs in Biomedicine 64 (2) (2001) 77 – 85. doi:https://doi.org/10.1016/S0169-2607(00)00093-6.

[6] H. Courtecuisse, J. Allard, P. Kerfriden, S. P. Bordas, S. Cotin, C. Duriez, Real-time simulation of contact and cutting of heterogeneous soft-tissues, Medical Image Analysis 18 (2) (2014) 394 – 410.

[7] H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D. Y. Lee, S. Cotin, GPU-based real-time soft tissue deformation with cutting and haptic feedback, Progress in Biophysics and Molecular Biology 103 (2) (2010)

159 – 168, special Issue on Biomechanical Modelling of Soft Tissue Motion. doi:https://doi.org/10.1016/j.pbiomolbio.2010.09.016.

[8] S. Niroomandi, I. Alfaro, D. Gonzlez, E. Cueto, F. Chinesta, Real-time simulation of surgery by reduced-order modeling and x-fem techniques, International Journal for Numerical Methods in Biomedical Engineering 28 (5) (2012) 574–588. doi:10.1002/cnm.1491.

[9] C. Quesada, D. Gonzlez, I. Alfaro, E. Cueto, F. Chinesta, Computational vademecums for real-time simulation of surgical cutting in haptic environments, International Journal for Numerical Methods in Engineering 108 (10) (2016) 1230–1247. doi:10.1002/nme.5252.

[10] S. Misra, K. Macura, K. Ramesh, A. Okamura, The importance of organ geometry and boundary constraints for planning of medical interventions, Medical Engineering & Physics 31 (2) (2009) 195 – 206. doi:https://doi.org/10.1016/j.medengphy.2008.08.002.

[11] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method, Computer Methods in Applied Mechanics and Engineering 199 (41) (2010) 2680–2686. doi:10.1016/j.cma.2010.05.011.

[12] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method, Applied Numerical Mathematics 62 (4) (2012) 328–341.

[13] E. Burman, S. Claus, P. Hansbo, M. G. Larson, A. Massing, CutFEM: Discretizing geometry and partial differential equations, International Journal for Numerical Methods in Engineering 104 (7) (2015) 472–501. doi:10.1002/nme.4823.

[14] T. Belytschko, T. Black, Elastic crack growth in finite elements with minimal remeshing, International Journal for Numerical Methods in Engineer-

ing 45 (5) (1999) 601–620. doi:10.1002/(SICI)1097-0207(19990620)45:5<601::AID-NME598>3.0.CO;2-S.

[15] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, International Journal for Numerical Methods in Engineering 46 (1) (1999) 131–150. doi:10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J.

[16] T. Strouboulis, I. Babuka, K. Copps, The design and analysis of the generalized finite element method, Computer Methods in Applied Mechanics and Engineering 181 (1) (2000) 43 – 69. doi:https://doi.org/10.1016/S0045-7825(99)00072-9.

[17] G. Tabor, P. G. Young, T. B. West, A. Benattayallah, Mesh construction from medical imaging for multiphysics simulation: Heat transfer and fluid flow in complex geometries, Engineering Applications of Computational Fluid Mechanics 1 (2) (2007) 126–135. arXiv:http://dx.doi.org/10.1080/19942060.2007.11015187, doi:10.1080/19942060.2007.11015187.

[18] M. Moumnassi, S. Belouettar, E. Béchet, S. P. Bordas, D. Quoirin, M. Potier-Ferry, Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces, Computer Methods in Applied Mechanics and Engineering 200 (5) (2011) 774 – 796. doi:https://doi.org/10.1016/j.cma.2010.10.002.

[19] G. F. Moita, M. A. Crisfield, A finite element formulation for 3-D continua using the co-rotational technique, International Journal for Numerical Methods in Engineering 39 (22) (1996) 3775–3792. doi:10.1002/(SICI)1097-0207(19961130)39:22<3775::AID-NME23>3.0.CO;2-W.

[20] S. Suwelack, S. Röhl, R. Dillmann, A.-L. Wekerle, H. Kenngott, B. Müller-Stich, C. Alt, S. Speidel, Quadratic Corotated Finite Elements for Real-

Time Soft Tissue Registration, Springer New York, New York, NY, 2012, pp. 39–50. doi:10.1007/978-1-4614-3172-5_6.

[21] N. Haouchine, J. Dequidt, I. Peterlik, E. Kerrien, M. O. Berger, S. Cotin, Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery, in: 2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2013, pp. 199–208. doi:10.1109/ISMAR.2013.6671780.

[22] J. Dequidt, E. Coevoet, L. Thins, C. Duriez, Vascular Neurosurgery Simulation with Bimanual Haptic Feedback, in: F. Jaillet, F. Zara, G. Zachmann (Eds.), Workshop on Virtual Reality Interaction and Physical Simulation, The Eurographics Association, 2015. doi:10.2312/vriphys.20151337.

[23] K. Sase, A. Fukuhara, T. Tsujita, A. Konno, GPU-accelerated surgery simulation for opening a brain fissure, ROBOMECH Journal 2 (1) (2015) 17. doi:10.1186/s40648-015-0040-0.

[24] F. Morin, H. Courtecuisse, I. Reinertsen, F. L. Lann, O. Palombi, Y. Payan, M. Chabanas, Brain-shift compensation using intraoperative ultrasound and constraint-based biomechanical simulation, Medical Image Analysis 40 (Supplement C) (2017) 133 – 153. doi:https://doi.org/10.1016/j.media.2017.06.003.

[25] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, S. Cotin, SOFA: A Multi-Model Framework for Interactive Physical Simulation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 283–321. doi:10.1007/8415_2012_125.

[26] A. K. Chopra, Dynamics of Structures, Prentice-Hall international series in civil engineering and engineering mechanics, Pearson Education, 2007.

[27] H. P. Bui, S. Tomar, H. Courtecuisse, S. Cotin, S. P. A. Bordas, Real-time error control for surgical simulation, IEEE Transactions on Biomedical Engineering 65 (3) (2018) 596–607. doi:10.1109/TBME.2017.2695587.

[28] H. P. Bui, S. Tomar, H. Courtecuisse, M. Audette, S. Cotin, S. P. Bordas, Controlling the error on target motion through real-time mesh adaptation: Applications to deep brain stimulation, International Journal for Numerical Methods in Biomedical Engineering e2958–n/aE2958 cnm.2958. doi:10.1002/cnm.2958.
URL http://dx.doi.org/10.1002/cnm.2958

[29] O. Zienkiewicz, R. Taylor, The Finite Element Method: Solid mechanics, Referex collection.Mecánica y materiales, Butterworth-Heinemann, 2000.

[30] G. R. Liu, S. S. Quek, Chapter 3 - Fundamentals for Finite Element Method, in: G. R. Liu, , S. S. Quek (Eds.), The Finite Element Method (Second Edition), second edition Edition, Butterworth-Heinemann, Oxford, 2014, pp. 43–79. doi:http://dx.doi.org/10.1016/B978-0-08-098356-1.00003-5.

[31] O. Zienkiewicz, R. Taylor, J. Zhu, The finite element method: Its basis and fundamentals, Vol. 1, Elsevier, 2013.

[32] C. Felippa, B. Haugen, A unified formulation of small-strain corotational finite elements: I. theory, Computer Methods in Applied Mechanics and Engineering 194 (2124) (2005) 2285 – 2335.

[33] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, B. Cutler, Stable real-time deformations, in: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2002, pp. 49–54.

[34] D. Baraff, A. Witkin, Large steps in cloth simulation, in: Proceedings of SIGGRAPH, 1998, pp. 43–54.

[35] S. Timoshenko, History of Strength of Materials: With a Brief Account of the History of Theory of Elasticity and Theory of Structures, Dover Civil and Mechanical Engineering Series, Dover Publications, 1953.

[36] F. Sotiropoulos, X. Yang, Immersed boundary methods for simulating fluidstructure interaction, Progress in Aerospace Sciences 65 (2014) 1–21. doi:10.1016/j.paerosci.2013.09.003.

[37] F. Qin, J. Chen, Z. Li, M. Cai, A Cartesian grid nonconforming immersed finite element method for planar elasticity interface problems, Computers & Mathematics with Applications 73 (3) (2017) 404–418. doi:10.1016/j.camwa.2016.11.033.

[38] J. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 1999.

[39] T.-P. Fries, S. Omerovi, Higher-order accurate integration of implicit geometries, International Journal for Numerical Methods in Engineering 106 (5) (2016) 323–371. doi:10.1002/nme.5121.

[40] N. M. Grosland, R. Bafna, V. A. Magnotta, Automated hexahedral meshing of anatomic structures using deformable registration, Computer Methods in Biomechanics and Biomedical Engineering 12 (1) (2009) 35–43, pMID: 18688764. arXiv:http://dx.doi.org/10.1080/10255840802136143, doi:10.1080/10255840802136143.

[41] G. Dhatt, G. Touzot, E. Lefranois, Finite Element Method, John Wiley & Sons, Inc., 2012. doi:10.1002/9781118569764.

[42] R. Plantefève, I. Peterlik, N. Haouchine, S. Cotin, Patient-specific biomechanical modeling for guidance during minimally-invasive hep-

atic surgery, Annals of Biomedical Engineering 44 (1) (2016) 139–153. doi:10.1007/s10439-015-1419-z.

[43] H. P. Bui, S. Tomar, H. Courtecuisse, M. Audette, S. Cotin, S. P. Bordas, Controlling the error on target motion through real-time mesh adaptation: Applications to deep brain stimulation, International Journal for Numerical Methods in Biomedical Engineering e2958–n/aE2958 cnm.2958. doi:10.1002/cnm.2958.

[44] T. Rabczuk, R. Gracie, J.-H. Song, T. Belytschko, Immersed particle method for fluidstructure interaction, International Journal for Numerical Methods in Engineering 81 (1) (2010) 48–71. doi:10.1002/nme.2670.

[45] M. Moumnassi, S. Bordas, R. Figueredo, P. Sansen, Analysis using higher-order xfem: implicit representation of geometrical features from a given parametric representation, Mechanics & Industry 15 (5) (2014) 443–448. doi:10.1051/meca/2014033.

[46] T. Hughes, J. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39) (2005) 4135 – 4195. doi:https://doi.org/10.1016/j.cma.2004.10.008.

[47] C. Anitescu, M. N. Hossain, T. Rabczuk, Recovery-based error estimation and adaptivity using high-order splines over hierarchical T-meshes, Computer Methods in Applied Mechanics and Engineering 328 (2018) 638 – 662. doi:https://doi.org/10.1016/j.cma.2017.08.032.