INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN BIOMEDICAL ENGINEERING
Int. J. Numer. Meth. Biomed. Engng. 2016; **00**:1–25
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cnm

Haptic simulation of tissue tearing during surgery

C. Quesada[1], I. Alfaro[1], D. González[1], F. Chinesta[2], E. Cueto[1,*]

[1] *Aragón Institute of Engineering Research. Universidad de Zaragoza. Zaragoza, Spain.*
[2] *Institute of High Performance Computing, ICI. Ecole Centrale de Nantes. Nantes, France.*

SUMMARY

We present a method for the real-time, interactive simulation of tissue tearing during laparoscopic surgery. The method is designed to work at haptic feedback rates (i.e., around 1kHz). Tissue tearing is simulated under the general framework of continuum damage mechanics. The problem is stated as a general, multidimensional parametric problem, which is solved by means of Proper Generalized Decomposition (PGD) methods. One of the main novelties is the reduction of history-dependent problems, such as damage mechanics, by resorting to an approach in which a reduced-order field of initial damage values is considered as a parameter of the formulation. We focus on the laparoscopic cholecystectomy procedure as a general example of the performance of the method.

Received . . .

KEY WORDS: Computational surgery; real-time simulation; tissue tearing; Proper Generalized Decomposition.

1. INTRODUCTION

Virtual reality including haptic rendering in the framework of computational surgery continues to be a challenging field of research despite nowadays computer capabilities [1, 2, 3, 4, 5, 6, 7, 8]. At the very heart of these difficulties lies the need for extremely fast feedback rates if physically realistic haptic rendering is intended. For a force to appear realistic to the sense of touch of the practitioner, feedback rates of some 500 Hz to 1 kHz are needed. This means simply that a complex non-linear solid and/or fluid mechanics problem needs to be solved some one thousand times per second so as to feed the haptic peripheral and that the perceived force appears realistic to the user.

In this framework many methods have been proposed in the literature to overcome such stringent requirements. Starting by naive condensation techniques for linear problems [9] to modern fully non-linear alternatives based upon explicit [5] or implicit finite elements [10, 11], several alternatives have been explored in our community.

Trying to bring state-of-the-art constitutive models for soft living tissues into play, our group has developed in recent years an alternative approach to the problem by employing model order reduction (MOR) techniques [12, 13]. These techniques allow for a substantial reduction in the

*Correspondence to: Elías Cueto. Universidad de Zaragoza. Edificio Betancourt. María de Luna, s.n. E-50018 Zaragoza, Spain.

Contract/grant sponsor: Spanish Ministry of Economy and Competitiveness. Regional Government of Aragon; contract/grant number: This work has been supported by the Spanish Ministry of Economy and Competitiveness through Grants number CICYT DPI2014-51844-C2-1-R and DPI2015-72365-EXP and by the Regional Government of Aragon and the European Social Fund, research group T88.

number of degrees of freedom of the model [14], while allowing for the use of energy and momentum conserving time integration schemes, for instance [15].

Particular attention has been paid, again due to the inherent difficulty of the problem, to the simulation of cutting processes during surgery. These usually involve the modification of the underlying mesh topology, therefore severely compromising the mentioned feedback rates [16, 17, 18, 19, 20]. Trying to avoid any mesh modification, X-FEM techniques have been employed in a recent number of works [21, 22]. However, during laparoscopic surgery, cutting plays a very limited role (specially in cholecystectomy, for instance, where only one procedure can be properly described as *cutting*). Instead, tissue tearing and ripping is much more common. Therefore, other alternatives for the description of such physical phenomena have been considered recently. For instance, approaches based upon fracture mechanics [23] or on averaged maximum principal stresses criteria [24] have been developed.

Thus, for a realistic simulation of surgery (and, particularly, laparoscopic cholecystectomy) it is of utmost importance to be able to reproduce all the tearing processes taking place. For instance, prior to performing the single cut that will separate the gall bladder from the cystic duct, the so-called Calot's (or hepatobiliary) triangle must be dissected so as to unveil the cystic duct and the cystic artery. During this process, fat tissue needs to be removed so as to obtain a view of the underlying structures [25]. This removal is made by simply tearing the fat tissue with the help of laparoscopic graspers. This is one of the most time-consuming procedures during cholecystectomy.

The objective of this work is to develop appropriate numerical techniques for the real-time simulation of tearing procedures during laparoscopic surgery. To that end, we work in the framework of the I3A surgery simulator developed by our group [26]. This simulator employs a particular architecture. It is based on the off-line construction of a sort of response surface or *computational vademecum* [27] for the particular type of surgery being simulated. This response surface constitutes in fact a sort of multi-dimensional solution to a parametric problem. The key ingredient, therefore, consists in selecting the most relevant parameters of the procedure to be reproduced, and to efficiently store in memory this high-dimensional parametric solution. Parameters are of very different nature: from boundary conditions (forces, contact conditions, ...) to material parameters, initial conditions, etc.

To solve such a high-dimensional problem, some kind of model order reduction is mandatory: one can not simply "mesh" the entire parametric or phase space and solve by means of finite elements or similar numerical strategies. This brute force approach is prevented by the so-called *curse of dimensionality*. In other words, by the exponential growth in the number of degrees of freedom with the number of dimensions of the problem.

To overcome the curse of dimensionality and, at the same time, allow for a very efficient storage of the response surface (or computational vademecum), we have employed Proper Generalized Decompositions (PGD) [28] [29] [30]. These will be briefly reviewed in Section 2 for completeness, although the interested reader can consult the previous references.

Section 3 presents the implemented description of soft living tissue tearing during laparoscopic surgery. It is based upon a very classical continuum damage mechanics approach [31, 32]. Section 4 presents the basics of the PGD implementation of the damage problem so as to obtain (off-line) a vademecum able to provide results (on-line) under real-time constraints. Section 5 presents some academic as well as practical examples of performance of the proposed method. The paper ends with a brief discussion on the performance of the method.

2. A HAPTIC SURGERY SIMULATOR BASED ON PGD

As mentioned in the Introduction, our surgery simulator is based upon a very classical principle: to compute off-line as many results as possible so as to minimize the on-line computational cost. This principle traces back to very classical simulators [9]. This approach is in principle limited to a purely linear description of the tissues. However, in our approach we generalize it by assuming that the displacement at any point of the tissue is given by a sort of non-linear response surface or

DOI: 10.1002/cnm

computational vademecum [27]:

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{s}, t, p_1, \dots, p_m), \tag{1}$$

where $\boldsymbol{x}$ represents the physical coordinates of the point under consideration, $\boldsymbol{s}$ represents the position of contact between the surgical instrument and the organ, t represents time, and, in general, $p_1, \dots, p_m$ represent a set of parameters involved in the description of the process. These can be, for instance, material parameters, boundary conditions, etc. In order to obtain a finite element (piece-wise polynomial) approximation of such an expression, PGD assumes a separated representation of the solution, i.e.,

$$u_j^n(\boldsymbol{x}, \boldsymbol{s}, t, p_1, \dots, p_m) = \sum_{k=1}^{n} X_j^k(\boldsymbol{x}) \cdot Y_j^k(\boldsymbol{s}) \cdot T_j^k(t) \cdot F_j^{k,1}(p_1) \cdot \ldots \cdot F_j^{k,m}(p_m), \tag{2}$$

where u_j refers to the j-th component of the displacement vector, $j = 1, 2, 3$, and functions $\boldsymbol{X}^k$, $\boldsymbol{Y}^k$, ..., represent the separated functions used to approximate the unknown field. These functions are in turn approximated in a finite element sense. Their precise form is determined by first employing a greedy algorithm, i.e., assumed that an approximation up to n terms is already known, see Eq. (2), we compute an improvement of this approximation, of the form

$$u_j^{n+1}(\boldsymbol{x}, \boldsymbol{s}, t, p_1, \dots, p_m) = u_j^n + X_j^{k+1}(\boldsymbol{x}) \cdot Y_j^{k+1}(\boldsymbol{s}) \cdot T_j^{k+1}(t) \cdot F_j^{k+1,1}(p_1) \cdot \ldots \cdot F_j^{k+1,m}(p_m).$$

Determining these new functions, indeed a product of functions, is obviously a non linear problem. These are, of course, interpolated by finite elements, so that we look for their nodal values. In principle any linearization procedure is valid (Newton, for instance), but most of our previous works we employ a fixed point strategy. Even if the solution is not guaranteed, as it is well known, this very simple procedure renders very good results in general. This is an already classical issue within PGD methods that can be consulted in several books [30] [29] or reviews [7].

In general, the number of terms n to be included in the approximation can be decided with the help of error estimators, possibly based on quantities of interest [33, 34]. The influence on the complexity of the resulting approach as a function of the number of parameters has also been studied since the early times of the PGD technique [35]. In general, it is well-known that the number of parameters increases the complexity of the solution linearly, instead of exponentially, as would be the case for a brute-force approach to the problem.

For physical phenomena depending on the history, such as for instance the general framework of dynamics of the surgery procedure, we can resort to expressing the displacement field in an incremental way (we deliberately omit here the parametric dependence for simplicity of the exposition):

$$\boldsymbol{u}^{n+1}(\boldsymbol{x}, t + \Delta t, \boldsymbol{u}^t, \boldsymbol{v}^t) = \boldsymbol{u}^n + \boldsymbol{R}(\boldsymbol{x}) \circ \boldsymbol{S}(\boldsymbol{u}^t) \circ \boldsymbol{T}(\boldsymbol{v}^t) \circ \boldsymbol{d}(t),$$

where the sought displacement field is now a function (as obviously corresponds to an initial and boundary-value problem) of the initial conditions, seen as the converged displacement $\boldsymbol{u}$ and velocity $\boldsymbol{v}$ fields of the previous time increment. More details on the PGD formulation of hyperelastodynamics can be found in [15].

One of the most important problems of model order reduction is that of solving non-linear problems. This is particularly noteworthy for biological soft tissues, of course. Since the obtaining of reduced tangent stiffness matrices generally involves the re-construction of the full-order ones, virtually any advantage of this type of techniques is lost. These drawbacks have been overcome by employing different strategies. In the field of PGD techniques several alternatives exist. For instance, in [36] an approach was presented that employed an explicit linearization of non-linear terms. Although formally complex due to the number of terms of the approximation, this approach has revealed to be very robust and useful. Still a fully nonlinear implicit approximations by employing asymptotic expansions was presented in [12]. This approach made use of the so-called Asymptotic Numerical Method [37] so as to solve a sequence of linear problems with the same tangent operator instead of iterating with different stiffness matrices. Still a third possibility exists, that has been

DOI: 10.1002/cnm

also adopted in other model order reduction approaches: that of employing the so-called empirical interpolation method [38]. This technique consists, essentially, in interpolating the non-linear term of the problem at a selected number of carefully chosen points. An in-deep analysis of all of the existing possibilities exceeds the scope of this paper, but nevertheless demonstrates that non-linearities no longer constitute a major difficulty in the field of model order reduction techniques. Noteworthy, contact detection can also be formulated in this framework by employing level set (distance field) techniques and a pointshell description of the slave body in the contact sequence [13].

Note that the assumed separate approximation of the unknown field, Eq. (2) provides a very efficient method not only for the off-line computation of the response surface itself —this is what Eq. (2) represents, in fact—, but for the very efficient storage of this surface in memory, in the form of vectors of nodal values for each separate function F_j^i. The particular solution is therefore never simulated, but evaluated in real time, the on-line part of the method. Therefore, unlike previous examples of surgery simulators such as [39], for instance, there is no need to establish multiple threads in the simulation, nor establishing different feedback requirements for the different tasks in the simulator.

Rendering, in general, is done under weaker requirements, usually in the order of some 30 Hz for a continuous perception of motion. In the rendering process the computation of nodal normal vectors at the deformed configuration of the solid is mandatory for an appropriate visualization of textures. Even these normals could be pre-computed and stored in memory in the spirit of Eq. (1). However, our prototype showed that they can be computed in runtime without interfering with the main loop of haptic response. All these tests were made on a HP ProBook 6470b laptop (Intel Core i7, with 8 Gb DDR3 PC3-12800 SDRAM), see Fig. 1.

Figure 1. Appearance of the developed simulator.

3. A CONTINUUM DAMAGE MECHANICS DESCRIPTION OF TISSUE TEARING

For some types of surgery, such as radial keratotomy, for instance, the key phenomena to simulate is cutting (which is performed with a diamond knife). In such a surgery, there is a neat cut or crack, with the appearance of two new surfaces (the crack lips) that could modify the topology

DOI: 10.1002/cnm

of the model. However, during cholecystectomy, for instance, the vast majority of the procedure is a cleaning process, where fat and fibrous tissue is eliminated so as to unveil the underlaying anatomical structures where to perform a single cut. Except from this final cut, the relevant physical phenomena taking place during cholecystectomy is more a damage process than cutting. Cutting in the framework of PGD has been previously analyzed in [40] and successfully incorporated to our simulator. Therefore, it falls outside the objective of this work. Instead, we focus on the description, and subsequent solution in a PGD framework, of tearing phenomena under a continuum damage mechanics approach.

3.1. A brief overview of continuum damage mechanics

The use of a damage model in this approach is justified by the fact that the torn tissue exhibits a behavior which is similar to damaged structures at a macro-scale level. Indeed, some surgical interventions require the use of instruments for scraping and removing pieces of tissue (e. g. generally adipose tissue) by tearing them apart to facilitate the access to certain organs (Calot's triangle in the case of cholecystectomy). These instruments generate a continuous increasing degradation of the tissue which eventually breaks apart. Therefore, an approach based on damage mechanics can be considered valid for simulating this process of tissue degradation.

There are several quantities, such as stress, strain, the strain energy, or the porosity of the material, that can be used to describe a mechanical representation of damage [41]. A commonly used quantity to describe damage, first defined in [42], is the relative area of micro-cracks and voids in any plane of the model oriented by its normal $\boldsymbol{n}$,

$$D = \frac{A_D}{A},$$

where A_D is the area of micro-cracks and voids, and A is the total area of the cross-section. If micro-stress concentrations and defect interactions are handled carefully, this damage variable D is well suited for continuum mechanics [41]. For this reason, this is the definition of damage variable used in the presented approach.

When damage is considered as isotropic (i. e.cracks and voids are distributed uniformly in all directions), then D is a scalar value; otherwise, the damage variable depends on the orientation $\boldsymbol{n}$, and a damage vector or tensor is required. For adipose tissue, isotropic damage could be a reasonable choice. For fiber-reinforced tissues such as vascular tissue, an anisotropic damage model should be preferred. Allard and coworkers [43], for instance, employ a fracture criterion to study tearing based on fracture mechanics that includes the fibrous character of the tissue.

The scalar values in the isotropic case range from zero to one, where $D = 0$ characterises an undamaged (virgin) state, $D < 1$ represents the initiation of a macro-crack, and $D = 1$ describes a fully damaged state.

Although we follow the classical Simó description of damage [32], we review it briefly for completeness. Consider a representative volume element of an isotropic damaged material loaded by a force $\boldsymbol{F}$. Since no micro-forces act on the surfaces of micro-cracks or micro-voids (represented by A_D), it is convenient to introduce an effective stress $\tilde{\boldsymbol{\sigma}}$ related to the surface that effectively resists the load,

$$\tilde{A} = A - A_D = A(1 - D).$$

By taking $\boldsymbol{F} = \boldsymbol{\sigma}\boldsymbol{n}A = \tilde{\boldsymbol{\sigma}}\boldsymbol{n}\tilde{A}$, the following equation for the effective stress $\tilde{\boldsymbol{\sigma}}$ is obtained:

$$\tilde{\boldsymbol{\sigma}} = \frac{\boldsymbol{\sigma}}{1 - D}.$$

The principle of strain equivalence assumes that any constitutive equation of strain for a damaged material may be derived by the constitutive laws of a virgin material, with the exception that the usual stress is replaced by the effective stress, see Fig. 2. For example, the linear elastic law of a damaged material can be written as follows:

$$\boldsymbol{\varepsilon} = \frac{\tilde{\boldsymbol{\sigma}}}{\mathbf{C}} = \frac{\boldsymbol{\sigma}}{(1 - D)\,\mathbf{C}}.$$

DOI: 10.1002/cnm

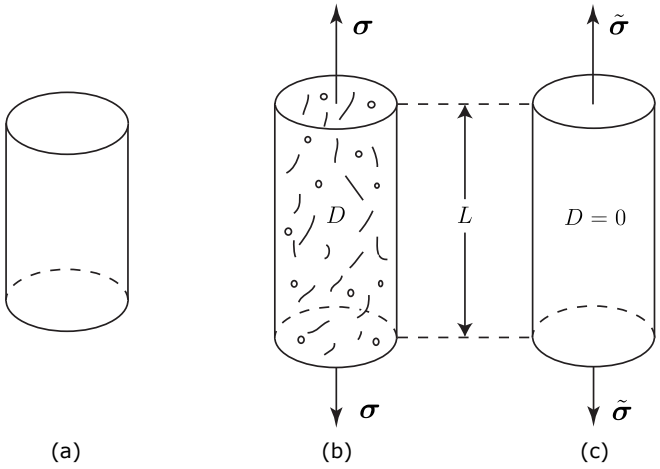

Figure 2. Different configurations of the damaged material: (a) virgin material, (b) damaged material, (c) equivalent virgin material.

This statement constitutes a non-rigorous hypothesis which has been demonstrated only in some particular cases of damage. However, its simplicity allows to establish a good working basis for elasticity or plasticity [31, 41].

Within the hypothesis of small strains and small displacements, the state variables that are taken into account are the elastic strain tensor $\boldsymbol{\varepsilon}$, and the damage variable D. Other state variables can be added to complete the model of damage, such as the temperature, but for simplicity they are not considered in this initial approach.

A state potential, i. e., a general stored energy function $\boldsymbol{\Psi}$ that allows to define the power involved in each physical process by means of variables associated with the state variables and state laws, can be written in function of the state variables, $\boldsymbol{\Psi}(\boldsymbol{\varepsilon}, D)$.

Thus, in terms of energy, each state variable has a corresponding associated variable. The associated variable of $\boldsymbol{\varepsilon}$ is the Cauchy stress tensor $\boldsymbol{\sigma}$, while the associated variable of D is usually defined as the generalized thermodynamic force $\bar{Y}$. Since D is dimensionless, and the product $-\bar{Y}\dot{D}$ is the power dissipated in the process of damage, $-\bar{Y}$ is seen as a volume energy density.

Assuming that the state laws are derived from a state potential, they can be obtained as

$$\boldsymbol{\sigma} = \frac{\partial \boldsymbol{\Psi}}{\partial \boldsymbol{\varepsilon}} \qquad \text{and} \qquad \bar{Y} = \frac{\partial \boldsymbol{\Psi}}{\partial D}. \tag{3}$$

The evolution of the damage described in this approach is based on the three-dimensional isotropic damage model developed by Simó in [32], which is, as he mentioned, well suited for computer implementations. Within this framework, the major assumption is that the damage process is totally controlled by the maximum strain achieved by the model up to the present time T.

To characterise the evolution of the damage variable D, it is necessary to introduce the strain energy of the undamaged material, $\boldsymbol{\Psi}_0$, as a scalar measure of the maximum strain reached during the load path. Thus, an equivalent strain energy can be defined by the expression

$$\Phi_t = \sqrt{2\,\boldsymbol{\Psi}_0\left(\boldsymbol{\varepsilon}(t)\right)}\,,$$

where $\boldsymbol{\varepsilon}(t)$ is the strain tensor at time t. Then, let Φ^m be the maximum value of Φ_t over the past history till the current time T, i. e.,

$$\Phi^m = \max_{t\in(-\infty,\,T]} \sqrt{2\,\boldsymbol{\Psi}_0\left(\boldsymbol{\varepsilon}(t)\right)}\,.$$

A damage criterion can be defined by the condition that at any time T of the loading process

$$\Phi_T \leq \Phi^m\,,$$

DOI: 10.1002/cnm

with Φ_T being the equivalent strain energy at current time, $\Phi_T = \sqrt{2\,\boldsymbol{\Psi}_0\left(\boldsymbol{\varepsilon}(T)\right)}$. The case in which $\Phi_T = \Phi^m$ and, simultaneously, a load is applied to a material already in a damaged state is particularly interesting, since this is the only situation for which the damage variable D evolves. This evolution is specified by the irreversible rate equation

$$\dot{D} = \begin{cases} h(\Phi, D)\,\dot{\Phi}, & \text{if } \Phi = \Phi^m \text{ and loading,} \\ 0, & \text{otherwise.} \end{cases}$$

Here, $h(\Phi, D)$ is a given function that characterises the damage process in the material. According to Simó [32], if h does not depend on D, the isotropic damage model can be expressed in the following equivalent form,

$$\boldsymbol{\sigma}(T) = g(\Phi^m)\,\frac{\partial \boldsymbol{\Psi}_0\left(\boldsymbol{\varepsilon}(T)\right)}{\partial \boldsymbol{\varepsilon}}, \tag{4}$$

with $g(\Phi^m)$ being a potential of dissipation. By applying the principle of strain equivalence to the stress tensor in Eq. (3), and comparing it with Eq. (4), it is clear that $g = 1 - D$, which can be followed by setting

$$h(\Phi) = -\frac{\mathrm{d}g(\Phi)}{\mathrm{d}\Phi}.$$

To determine the damage model, it is necessary to specify the potential of dissipation $g(\Phi^m)$ on the basis of experimental data. Simó proposed to adopt the following exponential form:

$$g(\Phi^m) = 1 - D = \beta + (1-\beta)\,\frac{1 - \mathrm{e}^{-\Phi^m/\alpha}}{\Phi^m/\alpha}, \tag{5}$$

where $\alpha \in [0,\, +\infty)$ and $\beta \in [0,\, 1]$ are given parameters. In consequence,

$$h(\Phi^m) = -\frac{\mathrm{d}g(\Phi^m)}{\mathrm{d}\Phi^m} = \frac{1-\beta}{(\Phi^m)^2}\,\left(\alpha - \mathrm{e}^{-\frac{\Phi^m}{\alpha}}(\alpha + \Phi^m)\right). \tag{6}$$

With all these definitions, it is possible to completely determine an expression for the stress tensor $\boldsymbol{\sigma}$. From Eq. (4), the expression of $\mathbf{C}$ becomes

$$\mathbf{C} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} = \frac{\partial}{\partial \boldsymbol{\varepsilon}}\left(g(\Phi^m)\frac{\partial \boldsymbol{\Psi}_0}{\partial \boldsymbol{\varepsilon}}\right) = g(\Phi^m)\frac{\partial^2 \boldsymbol{\Psi}_0}{\partial \boldsymbol{\varepsilon}^2} + \frac{\partial g(\Phi^m)}{\partial \boldsymbol{\varepsilon}}\,\frac{\partial \boldsymbol{\Psi}_0}{\partial \boldsymbol{\varepsilon}},$$

with

$$\frac{\partial \boldsymbol{\Psi}_0}{\partial \boldsymbol{\varepsilon}} = \boldsymbol{\sigma}_0 \qquad \text{and} \qquad \frac{\partial^2 \boldsymbol{\Psi}_0}{\partial \boldsymbol{\varepsilon}^2} = \mathbf{C}_0,$$

where $\boldsymbol{\sigma}_0$ and $\mathbf{C}_0$ are, respectively, the stress and the elasticity tensors when the material is undamaged. The remaining derivative expression can be calculated by means of the chain rule,

$$\frac{\partial g(\Phi^m)}{\partial \boldsymbol{\varepsilon}} = \frac{\partial g(\Phi^m)}{\partial \Phi^m}\,\frac{\partial \Phi^m}{\partial \boldsymbol{\Psi}_0}\,\frac{\partial \boldsymbol{\Psi}_0}{\partial \boldsymbol{\varepsilon}}.$$

Finally, the evolution of the stress tensor $\boldsymbol{\sigma}$ can be expressed as:

$$\boldsymbol{\sigma}(T) = \begin{cases} \left(g(\Phi^m)\,\mathbf{C}_0 - h'(\Phi^m)\,(\boldsymbol{\sigma}_0 \otimes \boldsymbol{\sigma}_0)\right) : \boldsymbol{\varepsilon}, & \text{if } \Phi_T = \Phi^m \text{ and loading,} \\ g(\Phi^m)\,\mathbf{C}_0 : \boldsymbol{\varepsilon}, & \text{otherwise,} \end{cases} \tag{7}$$

with

$$h'(\Phi^m) = \frac{1-\beta}{(\Phi^m)^3}\,\left(\alpha - \mathrm{e}^{-\frac{\Phi^m}{\alpha}}(\alpha + \Phi^m)\right).$$

DOI: 10.1002/cnm

3.2. Weak form of the problem

To solve the previous damage model in a PGD framework so as to render a fully interactive simulator, it is necessary to perform the simulation in a stepwise, incremental approach.

An explicit formulation for a three-dimensional linear elastic model is presented here, which is defined by the strain energy function

$$\boldsymbol{\Psi}_0(\boldsymbol{\varepsilon}) = \frac{1}{2}\lambda\,(\mathrm{tr}\,\boldsymbol{\varepsilon})^2 + \mu\boldsymbol{\varepsilon} : \boldsymbol{\varepsilon},$$

where λ and μ are the Lamé parameters.

To obtain an explicit formulation compatible with the real-time constraints, consider that the last step has been computed and the following step is sought. The weak form of the linear elasticity problem can be written as *find the displacement field* $\boldsymbol{u} \in \mathcal{H}^1(\Omega)$ *such that for all* $\boldsymbol{u}^* \in (H)^1_0(\Omega)$,

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{u}^*_{t+\Delta t} : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{u}_{t+\Delta t}\,\mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{u}^*_{t+\Delta t}\cdot\boldsymbol{f}_{t+\Delta t}\,\mathrm{d}\Gamma, \tag{8}$$

with

$$\mathbf{C} = \begin{cases} \left(g(\Phi^m)\,\mathbf{C}_0 - h'(\Phi^m)\,(\boldsymbol{\sigma}_0\otimes\boldsymbol{\sigma}_0)\right), & \text{if } \Phi_T = \Phi^m \text{ and loading,} \\ g(\Phi^m)\,\mathbf{C}_0, & \text{otherwise,} \end{cases}$$

and where the subscript $t+\Delta t$ denotes the following step, and $\boldsymbol{f}$ is the surface traction. $\boldsymbol{\nabla}_s = \frac{1}{2}(\boldsymbol{\nabla}+\boldsymbol{\nabla}^T)$ is the symmetric gradient operator; and $\Gamma_t = \Gamma_{t1}\cup\Gamma_{t2}$ are the regions of homogeneous and non-homogeneous natural boundary conditions, respectively. The displacement field $\boldsymbol{u}$ and applied force $\boldsymbol{f}$, for which the following step has to be computed, can be expressed as

$$\boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \Delta\boldsymbol{u} \qquad \text{and} \qquad \boldsymbol{f}_{t+\Delta t} = \boldsymbol{f}_t + \Delta\boldsymbol{f},$$

where $\boldsymbol{u}_t$ is already known from the previous step, and $\Delta\boldsymbol{u}$ is the new unknown to be solved for. Correspondingly, the admissible variation of $\boldsymbol{u}_{t+\Delta t}$ can be expressed as

$$\boldsymbol{u}^*_{t+\Delta t} = \Delta\boldsymbol{u}^*.$$

Thus, Eq. (8) can be re-written as

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\left(\boldsymbol{u}_t + \Delta\boldsymbol{u}\right)\,\mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^*\cdot\left(\boldsymbol{f}_t + \Delta\boldsymbol{f}\right)\,\mathrm{d}\Gamma,$$

which leads to

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{u}_t\,\mathrm{d}\Omega + \int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}\,\mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^*\cdot\boldsymbol{f}_t\,\mathrm{d}\Gamma + \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^*\cdot\Delta\boldsymbol{f}\,\mathrm{d}\Gamma. \tag{9}$$

Since $\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\boldsymbol{u}_t\,\mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^*\cdot\boldsymbol{f}_t\,\mathrm{d}\Gamma$ is the expression of equilibrium at the previous step, both terms cancel each other in Eq. (9). As a consequence, the incremental formulation can be finally expressed as

$$\int_{\Omega} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}\,\mathrm{d}\Omega = \int_{\Gamma_{t2}} \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^*\cdot\Delta\boldsymbol{f}\,\mathrm{d}\Gamma. \tag{10}$$

Standard finite element methods approximate Eq. (10), obtaining an algebraic expression $\boldsymbol{K}_t\cdot\Delta\boldsymbol{u} = \Delta\boldsymbol{f}$, where $\boldsymbol{K}_t$ represents the global stiffness matrix of the problem. However, standard FE algorithms are only valid for solving off-line problems rather than for real-time simulation. The main bottleneck is the need to compute at each step the inverse of the updated stiffness matrix $\boldsymbol{K}_t^{-1}$. Indeed, finding the inverse of a large matrix is a very time-consuming process, and its computation may take a very long time—seconds, or even minutes, depending on the size of the matrix and the power of the processor.

DOI: 10.1002/cnm

4. A REAL-TIME APPROACH TO THE CONTINUUM DAMAGE PROBLEM

As the last section made clear, an incremental FEM approach cannot solve by itself the simulation of damage evolution in a real-time framework. An approach based on PGD is presented in this section.

4.1. A PGD approach to the damage problem

The main modification of the scheme developed in the previous section consists in redefining the problem so that the initial conditions of each step, i. e., the scalar fields of damage (represented by their respective vectors of Gauss point values, $\boldsymbol{g}_n$ and $\boldsymbol{h}'_n$ at time step t_n) can be addressed as parameters for the subsequent computation at time step $n+1$, i. e., $\boldsymbol{g}_{n+1}$ and $\boldsymbol{h}'_{n+1}$. As in previous approaches, the position of the applied load $\boldsymbol{s}$ is also included as a parameter. The value of the load increment $\Delta\boldsymbol{f}$ can also be added as a fourth parameter but, for clarity of exposition, the same load increment is considered at each step in all the approaches described henceforth.

Formally, the multidimensional form of the displacement field will be defined in a phase space

$$\Delta\boldsymbol{u} : \Omega \times \mathcal{G}_1 \times \ldots \times \mathcal{G}_{\mathtt{ngp}} \times \mathcal{H}_1 \times \ldots \times \mathcal{H}_{\mathtt{ngp}} \times \bar{\Gamma} \to \mathbb{R}^3, \tag{11}$$

where $\mathcal{G}_i$ and $\mathcal{H}_i$ are the considered intervals of variation of each component of the vector of damage variables $\boldsymbol{g}_n$ and $\boldsymbol{h}'_n$, respectively, taken as initial conditions for the subsequent time step. $\mathcal{G}_i \in [0,1]$, and $\mathcal{H}_i \in [0,+\infty)$ for all $i = 1, 2, \ldots, \mathtt{ngp}$, with $\mathtt{ngp}$ being the number of Gauss integration points of the whole model.

In other words, considering the damage variables (actually, continuous scalar fields defined in the whole body under consideration) as parameters of the formulation implies considering their value at every Gauss point of the model as a parameter. For meshes of practical interest, the number of resulting parameters will be prohibitive even for PGD techniques. This problem will be addressed later.

For the time being, to obtain a parametric solution for any initial damage condition, within these intervals, the weak form of the problem, Eq. (10), has to be redefined as follows:

$$\int_{\bar{\Gamma}}\int_{\mathcal{H}_{\mathtt{ngp}}}\cdots\int_{\mathcal{H}_1}\int_{\mathcal{G}_{\mathtt{ngp}}}\cdots\int_{\mathcal{G}_1}\int_{\Omega}\mathcal{U}\;\mathrm{d}\Omega\,\mathrm{d}\mathcal{G}_1\cdots\mathrm{d}\mathcal{G}_{\mathtt{ngp}}\,\mathrm{d}\mathcal{H}_1\cdots\mathrm{d}\mathcal{H}_{\mathtt{ngp}}\,\mathrm{d}\bar{\Gamma} = \int_{\bar{\Gamma}}\int_{\Gamma_{t_2}}\Delta\boldsymbol{u}^*\cdot\Delta\boldsymbol{f}\,\mathrm{d}\Gamma\,\mathrm{d}\bar{\Gamma}, \tag{12}$$

where $\mathcal{U} = \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^* : \mathbf{C} : \boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}$, and $\mathbf{C}$ can be obtained from Eq. (3.2) as

$$\mathbf{C} = g\,\mathbf{C}_0 - h'\,(\boldsymbol{\sigma}_0 \otimes \boldsymbol{\sigma}_0)\,. \tag{13}$$

The PGD formulation of the problem for each component $j = 1, 2, 3$ of $\Delta\boldsymbol{u}$ can be expressed —the computation at time step n being converged— as

$$\Delta u_j^n(\boldsymbol{x}; g_1, \ldots, g_{\mathtt{ngp}}, h'_1, \ldots, h'_{\mathtt{ngp}}, \boldsymbol{s}) = \sum_{i=1}^{n} F_j^i(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathtt{ngp}} {G_K}_j^i(g_K) \cdot \prod_{K=1}^{\mathtt{ngp}} {H_K}_j^i(h'_K) \cdot J_j^i(\boldsymbol{s}). \tag{14}$$

Then, the expression for obtaining the $(n+1)$-th term can be written as

$$\Delta u_j^{n+1}(\boldsymbol{x}; g_1, \ldots, g_{\mathtt{ngp}}, h'_1, \ldots, h'_{\mathtt{ngp}}, \boldsymbol{s}) = \Delta u_j^n(\boldsymbol{x}; g_1, \ldots, g_{\mathtt{ngp}}, h'_1, \ldots, h'_{\mathtt{ngp}}, \boldsymbol{s}) + R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathtt{ngp}} S_{Kj}(g_K) \cdot \prod_{K=1}^{\mathtt{ngp}} T_{Kj}(h'_K) \cdot U_j(\boldsymbol{s}).$$

DOI: 10.1002/cnm

Lastly, the test function can be obtained by applying the rules of variational calculus to the expression above,

$$\begin{aligned}
\Delta u_j^*(\boldsymbol{x}; g_1, \ldots, g_{\mathtt{ngp}}, h'_1, \ldots, h'_{\mathtt{ngp}}, \boldsymbol{s}) = \\
&= R_j^*(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathtt{ngp}} S_{Kj}(g_K) \cdot \prod_{K=1}^{\mathtt{ngp}} T_{Kj}(h'_K) \cdot U_j(\boldsymbol{s}) + \\
&+ R_j(\boldsymbol{x}) \cdot S_{1j}^*(g_1) \cdot \ldots \cdot S_{\mathtt{ngp}j}(g_{\mathtt{ngp}}) \cdot \prod_{K=1}^{\mathtt{ngp}} T_{Kj}(h'_K) \cdot U_j(\boldsymbol{s}) + \ldots + \\
&+ R_j(\boldsymbol{x}) \cdot S_{1j}(g_1) \cdot \ldots \cdot S_{pj}^*(g_{\mathtt{ngp}}) \cdot \prod_{K=1}^{\mathtt{ngp}} T_{Kj}(h'_K) \cdot U_j(\boldsymbol{s}) + \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathtt{ngp}} S_{Kj}(g_K) \cdot T_{1j}^*(h'_1) \cdot \ldots \cdot T_{\mathtt{ngp}j}(h'_{\mathtt{ngp}}) \cdot U_j(\boldsymbol{s}) + \ldots + \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathtt{ngp}} S_{Kj}(g_K) \cdot T_{1j}(h'_1) \cdot \ldots \cdot T_{pj}^*(h'_{\mathtt{ngp}}) \cdot U_j(\boldsymbol{s}) + \\
&+ R_j(\boldsymbol{x}) \cdot \prod_{K=1}^{\mathtt{ngp}} S_{Kj}(g_K) \cdot \prod_{K=1}^{\mathtt{ngp}} T_{Kj}(h'_K) \cdot U_j^*(\boldsymbol{s}).
\end{aligned}$$

By substituting the approximations $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{u}^*$ into the weak form of the problem, Eq. (12), a computational vademecum is obtained for *any* value of the (discretized) damage field. Instead, $\Delta\boldsymbol{u}$ can be obtained almost automatically just by particularizing the computational vademecum with the values of $\boldsymbol{x}, g_1, \ldots, g_{\mathtt{ngp}}, h'_1, \ldots, h'_{\mathtt{ngp}}$, and $\boldsymbol{s}$ at any step. However, although it is possible to obtain a vademecum with a relatively high number of parameters, its computation when there exist hundreds of thousands (if not millions) of parameters presents major difficulties, as mentioned before. The next section is devoted to the reduction of this complexity.

4.2. Reducing the dimensionality of the field of initial conditions

Standard FE procedures compute the value of damage variables at Gauss points. As mentioned before, considering each one of these values (two per Gauss point, indeed) for meshes of hundreds of thousands of nodes remains prohibitive. Therefore, it seems reasonable to optimize somehow the parametrization of the space of initial conditions. To achieve this, an optimal basis is constructed using POD from the results of complete, full order, similar problems.

The POD decomposition provides an optimal basis (in a given norm) on which to project the results of any particular problem. The order of this basis can be selected to be as reduced as possible, depending on the assumed margin of error. Thus, reduced-order bases of the initial conditions can be re-injected into the PGD problem to obtain a computational vademecum, which takes as input parameters the projected coefficients of the scalar fields of damage of the previous time step.

In this particular case, the scalar fields of damage variables $\boldsymbol{g}$ and $\boldsymbol{h}'$ can be approximated, using the POD reduced-order basis, as

$$\boldsymbol{g} = \sum_{i=1}^{q} \xi_i \cdot \boldsymbol{\upsilon}_i \qquad \text{and} \qquad \boldsymbol{h}' = \sum_{j=1}^{r} \chi_j \cdot \boldsymbol{\rho}_j, \tag{15}$$

where $\boldsymbol{\upsilon}_i$ and $\boldsymbol{\rho}_j$ are the set of basis vectors of the new reduced-order basis; ξ_i and χ_j are the coefficients associated to each vector of the reduced basis; and q and r indicate the size of each reduced basis, with $q, r \ll \mathtt{ngp}$ (for practical purposes, not higher than ten), and not necessarily $q = r$. Vectors $\boldsymbol{\upsilon}_i$ and $\boldsymbol{\rho}_j$ are approximated using standard finite element procedures. Therefore, damage values at Gauss points can be obtained by standard projection.

DOI: 10.1002/cnm

With this approximations, the weak form of the parametric problem is now defined as:

$$\int_{\bar{\Gamma}}\int_{\mathcal{X}_r}\cdots\int_{\mathcal{X}_1}\int_{\mathcal{E}_q}\cdots\int_{\mathcal{E}_1}\int_{\Omega}\mathcal{U}\;\mathrm{d}\Omega\,\mathrm{d}\mathcal{E}_1\cdots\mathrm{d}\mathcal{E}_q\,\mathrm{d}\mathcal{X}_1\cdots\mathrm{d}\mathcal{X}_r\,\mathrm{d}\bar{\Gamma}$$
$$=\int_{\bar{\Gamma}}\int_{\mathcal{X}_r}\cdots\int_{\mathcal{X}_1}\int_{\mathcal{E}_q}\cdots\int_{\mathcal{E}_1}\int_{\Gamma_{t_2}}\Delta\boldsymbol{u}^*\cdot\Delta\boldsymbol{t}\,\mathrm{d}\Gamma\,\mathrm{d}\bar{\Gamma},\quad(16)$$

where $\mathcal{U}=\boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}^*:\mathbf{C}:\boldsymbol{\nabla}_{\mathrm{s}}\Delta\boldsymbol{u}$.

The PGD formulation of the solution $\Delta\boldsymbol{u}$ can be expressed, assuming the computation at iteration n of the procedure, as

$$\Delta u_j^n(\boldsymbol{x};\xi_1,\ldots,\xi_q,\chi_1,\ldots,\chi_r,\boldsymbol{s})=\sum_{i=1}^{n}F_j^i(\boldsymbol{x})\cdot\prod_{K=1}^{q}G_K{}_j^i(\xi_K)\cdot\prod_{L=1}^{r}H_L{}_j^i(\chi_L)\cdot J_j^i(\boldsymbol{s}).\quad(17)$$

Note that in Eq. (17) an abuse of notation has been made for simplicity. Functions $F_j^i, G_K{}_j^i, H_L{}_j^i$ are of course not the same as in Eq. (14), since the later versions are now functions of the reduced variables ξ_K and χ_L. Then, the expression for obtaining the $(n+1)$-th term can be written as

$$\Delta u_j^{n+1}(\boldsymbol{x};\xi_1,\ldots,\xi_q,\chi_1,\ldots,\chi_r,\boldsymbol{s})=\Delta u_j^n(\boldsymbol{x};\xi_1,\ldots,\xi_q,\chi_1,\ldots,\chi_r,\boldsymbol{s})$$
$$+R_j(\boldsymbol{x},\boldsymbol{\upsilon},\boldsymbol{\rho})\cdot\prod_{K=1}^{q}S_{Kj}(\xi_K)\cdot\prod_{L=1}^{r}T_{Lj}(\chi_L)\cdot U_j(\boldsymbol{s}).\quad(18)$$

Lastly, the test function can be obtained by applying the rules of variational calculus to the expression above,

$$\begin{aligned}\Delta u_j^*(\boldsymbol{x};\xi_1,\ldots,\xi_q,\chi_1,\ldots,\chi_r,\boldsymbol{s})&\\ =R_j^*(\boldsymbol{x})\cdot\prod_{K=1}^{q}S_{Kj}(\xi_K)\cdot\prod_{L=1}^{r}T_{Lj}(\chi_L)\cdot U_j(\boldsymbol{s})&\\ +R_j(\boldsymbol{x})\cdot S_{1j}^*(\xi_1)\cdot\ldots\cdot S_{qj}(\xi_q)\cdot\prod_{L=1}^{r}T_{Lj}(\chi_L)\cdot U_j(\boldsymbol{s})+\ldots&\\ +R_j(\boldsymbol{x})\cdot S_{1j}(\xi_1)\cdot\ldots\cdot S_{qj}^*(\xi_q)\cdot\prod_{L=1}^{r}T_{Lj}(\chi_L)\cdot U_j(\boldsymbol{s})&\\ +R_j(\boldsymbol{x})\cdot\prod_{K=1}^{q}S_{Kj}(\xi_K)\cdot T_{1j}^*(\chi_1)\cdot\ldots\cdot T_{rj}(\chi_r)\cdot U_j(\boldsymbol{s})+\ldots&\\ +R_j(\boldsymbol{x})\cdot\prod_{K=1}^{q}S_{Kj}(\xi_K)\cdot T_{1j}(\chi_1)\cdot\ldots\cdot T_{rj}^*(\chi_r)\cdot U_j(\boldsymbol{s})&\\ +R_j(\boldsymbol{x})\cdot\prod_{K=1}^{q}S_{Kj}(\xi_K)\cdot\prod_{L=1}^{r}T_{Lj}(\chi_L)\cdot U_j^*(\boldsymbol{s}).&\end{aligned}$$

Once the number of parameters of the problem has been reduced to only a few tens or less, both the development of the mathematical equations and the computation of the vademecum are clearly much simpler than when the approach based solely on the PGD is used. A sketch of the proposed method can be found in Algorithm 1.

The precise matrix form of the just presented Algorithm is included in Appendix A.

For the POD basis, our general criterion is to employ the number of modes able to reproduce the 95% of the energy of the system. In general, this can be improved without problems to some 99%. For the PGD ones, we usually employ a stopping criterion consisting of

DOI: 10.1002/cnm

initialisation: $t = 0$, $\boldsymbol{u}_t = \boldsymbol{0}$, $g_t^p = 1$, $h_t^p = 0$, $\Phi_t^p = 0$, $(\Phi^m)^p = 0$, $\boldsymbol{S}_t^p = \boldsymbol{0}$, for each Gauss point p ;
while *t has not reached the end of simulation* **do**
  determine $\Delta \boldsymbol{f}$ from Δt ;
  compute ξ_t by L_2-projection of $\boldsymbol{g_t}$ onto $\boldsymbol{\upsilon}$;
  compute χ_t by L_2-projection of $\boldsymbol{h_t}$ onto $\boldsymbol{\rho}$;
  particularise computational vademecum to obtain $\Delta \boldsymbol{u}$;
  $\boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \Delta u$;
  for *each Gauss point p* **do**
    $\boldsymbol{\varepsilon}^p_{t+\Delta t} = \boldsymbol{B} \cdot \boldsymbol{u}_{t+\Delta t}$;
    compute $\Psi^p_{t+\Delta t}$ and $\Phi^p_{t+\Delta t}$;
    if $\Phi^p_{t+\Delta t} > (\Phi^m)^p$ **then**
      $(\Phi^m)^p = \Phi^p_{t+\Delta t}$;
      compute new values of $g^p_{t+\Delta t}$ from Eq. (5) ;
      compute new values of $h^p_{t+\Delta t}$ from Eq. (6) ;
    else
      $g^p_{t+\Delta t} = g^p_t$, $h^p_{t+\Delta t} = 0$;
    end
    $(\boldsymbol{\sigma}_0)^p_{t+\Delta t} = \mathbf{C}_0 \cdot \boldsymbol{\varepsilon}^p_{t+\Delta t}$;
  end
  $t = t + \Delta t$;
end

Algorithm 1: Explicit formulation of the POD–PGD algorithm.

1. The fixed point iteration is stopped once the modulus of the functions does not suffer changes bigger than 10^{-4}.
2. The second loop (the number of PGD modes) is stopped once the modulus of the added functional product reaches values below 10^{-4}.

In general, complexity of PGD algorithms scales proportionally to the number of elements along each separated dimension times the number of modes times the number of fixed-point iterations. This complexity depends, of course, on the degree of separability of the solution (for an in-deep discussion on this see [44]) and the level of discretization of the physical as well as the parametric spaces. Therefore, this complexity scales roughly linearly with the number of dimensions (and not exponentially, which is typical of mesh-based methods). This constitutes the true advantage of employing PGD in this type of problems.

5. NUMERICAL EXAMPLES

To better illustrate how the proposed method behaves, some examples have been studied. In particular, Section 5.1 analyses some basic problems using a unit cube to test the validity of the approach, and Section 5.2 shows how the method can be applied to simulate the surgical removal of adipose tissue from the gallbladder.

5.1. Unit cube

A series of simple test examples have been developed to show the applicability of the method and to study the influence of the projection onto the POD basis procedure on the accuracy of the results. The model common to these simple test examples is a unit cube consisting of a single regular eight-node hexahedral element. The cube is fixed at its base, and different loads are applied on its upper

DOI: 10.1002/cnm

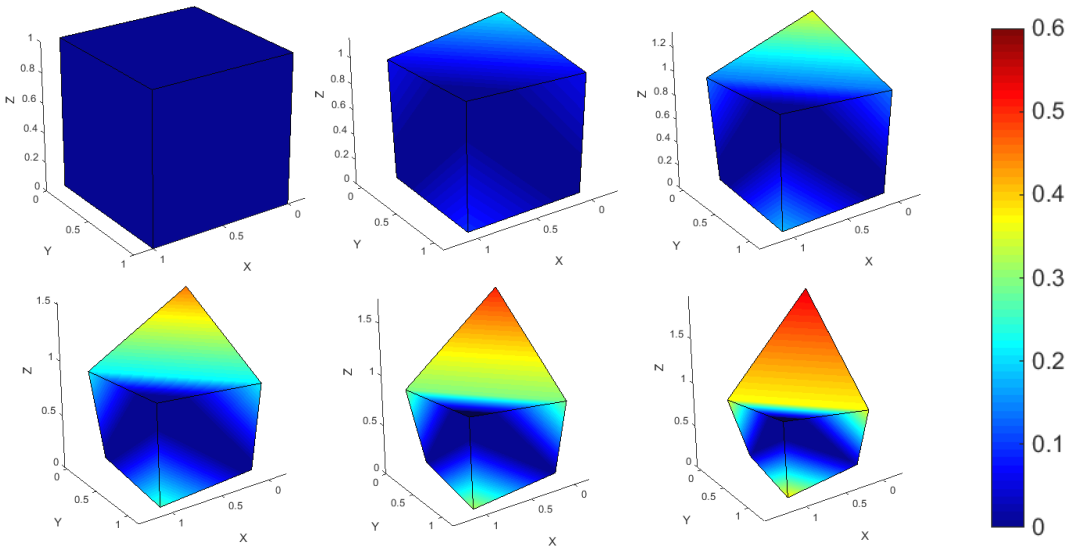

Figure 3. Damage evolution of a unit cube loaded at a node. From left to right and from top to bottom, initial configuration, and pseudo-time steps 2, 4, 6, 8, and 10 are depicted. Colours show the values of the interpolated variable of damage D.

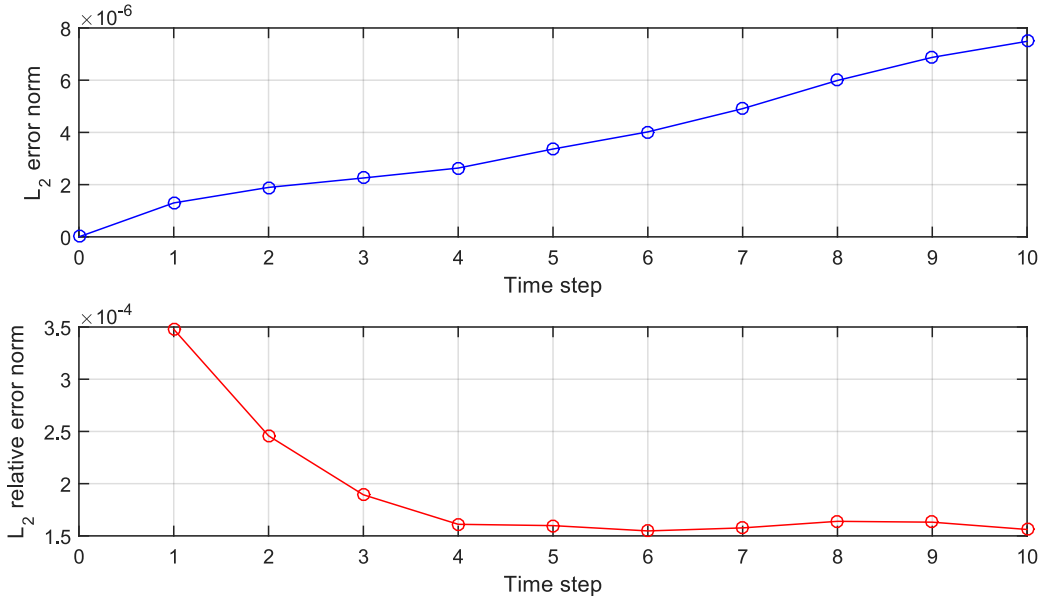

Figure 4. L_2 error norm (top) and L_2 relative error norm (bottom) of the unit cube loaded at a node for each of the 10 computed pseudo-time steps.

part. § 5.1.1 describes the damage evolution of a cube in which one of its upper corners is pulled upwards, while § 5.1.2 describes the damage evolution a cube undergoing pure shear deformation.

5.1.1. Load at a node A traction load is applied vertically to a single node of the upper part of the unit cube. One of the nodes of the base is fully fixed, whereas the remaining nodes of the base allow displacements along the $Z = 0$ plane. The magnitude of the applied load is constant over time, so each pseudo-time step Δt is associated to the same increment of force, which has a value of $\Delta \boldsymbol{f} = 0.05$ N in this example. The considered material is linear elastic, with a Young's modulus of

DOI: 10.1002/cnm

$E = 5$ Pa, and a Poisson's ratio of $\nu = 0.3$; and the parameters of damage are $\alpha = \beta = 0.4$, which are chosen so that the model can reach quickly an intermediate level of damage.

To obtain the POD reduced basis, a standard FEM model incorporating damage is taken as a reference model. The previously described load, boundary conditions, and material and damage parameters are applied. This reference model is simulated for 10 pseudo-time steps and, at each step, the scalar damage field $\boldsymbol{g}$ is recorded at the Gauss integration points of the element. In this preliminary example $h' = 0$ is assumed. Eight Gauss points (two per dimension, 2^3) are considered, thus obtaining ten different sets (one per step) of eight values of the damage field. Then, these ten sets are used as snapshots with which to compute a POD basis. Since each of the ten vectors of the resulting basis, provided by the POD, has an associated value of explained variance, the selected vectors to form the reduced basis $\boldsymbol{v}$ are those that account for a greater proportion of explained variance. The number of selected vectors should not be high either, so a balance must be maintained between both factors, which depend on each situation. In this example, the selected POD reduced basis $\boldsymbol{v}$ consists of three vectors (out of the mentioned ten vector for the full-order problem history), which explain nearly the 100% of the total variance of the provided snapshots.

Once the POD reduced basis $\boldsymbol{v}$ is known, a PGD computational vademecum can be obtained. To this end, it is necessary to determine the discretization of coefficients ξ_K (with $K = 1, 2, 3$), represented by matrices $\boldsymbol{G_K}$ in Appendix A. Since the solution of this example is already known (it is the previously computed reference model), the limits between which the coefficients ξ_K vary can be easily approximated by projecting the damage values of the reference model into the POD reduced basis. By doing this, the discretisation in this example can be defined by the following interval limits and number of uniformly distributed mesh nodes: $\xi_1 \in [-3, -2]$ using $4\,000$ nodes; $\xi_2 \in [-0.3, 0.3]$ using $2\,400$ nodes; and $\xi_3 \in [-0.1, 0.3]$ using $1\,600$ nodes. With this coefficient discretisation ξ_K, PGD required 28 modes before reaching convergence.

The computed vademecum can be incorporated into the explicit incremental POD–PGD algorithm (Algorithm 1). Again, ten pseudo-time steps are computed in real-time, six of which are shown in Fig. 3. In analysing the sequence, it can be observed that damage concentrates around the corner which is pulled upwards, and it intensifies as the magnitude of the load increases. L_2 error norms have been computed for each step (see Fig. 4), which allow to compare the differences between the results obtained by the POD–PGD algorithm and the reference solution.

5.1.2. Pure shear In this example, traction loads of the same value are applied horizontally to the upper four nodes of a unit cube to achieve pure shear deformation. The four nodes of the base are fully fixed and, unlike the previous example, the force increment vary over time in terms of Fig. 5. In total, 200 pseudo-time steps are simulated with the following values of force increment: $\Delta \boldsymbol{f} = 0.004$ from steps 1 to 25, $\Delta \boldsymbol{f} = -0.004$ from steps 26 to 50, $\Delta \boldsymbol{f} = 0.008$ from steps 51 to 75, $\Delta \boldsymbol{f} = -0.008$ from steps 76 to 100, $\Delta \boldsymbol{f} = 0.012$ from steps 101 to 125, $\Delta \boldsymbol{f} = -0.012$ from steps 126 to 150, $\Delta \boldsymbol{f} = 0.016$ from steps 151 to 175, and $\Delta \boldsymbol{f} = -0.016$ from steps 176 to 200. This allows the cube to return to the original configuration each 50 steps (conserving the gained damage, though), and to reach higher values of the total applied load $\boldsymbol{f}$ at each cycle.

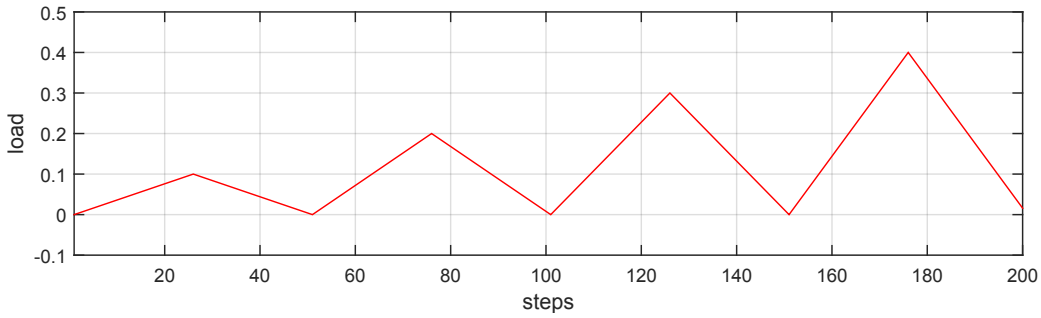

Figure 5. Total load $\boldsymbol{f}$ applied at each simulation step to a unit cube undergoing pure shear deformation.

DOI: 10.1002/cnm

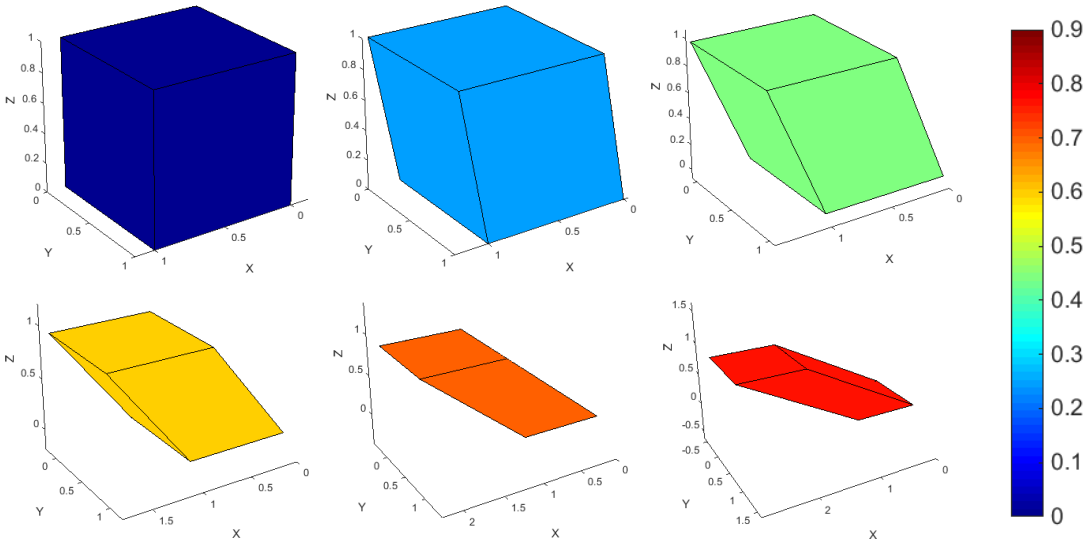

Figure 6. Damage evolution of a unit cube undergoing pure shear deformation. Initial configuration and steps 20, 70, 120, 170, and 175 are shown, which correspond to values of total applied force $\boldsymbol{f}$ of 0, 0.08, 0.16, 0.24, 0.32, and 0.4 N, respectively.

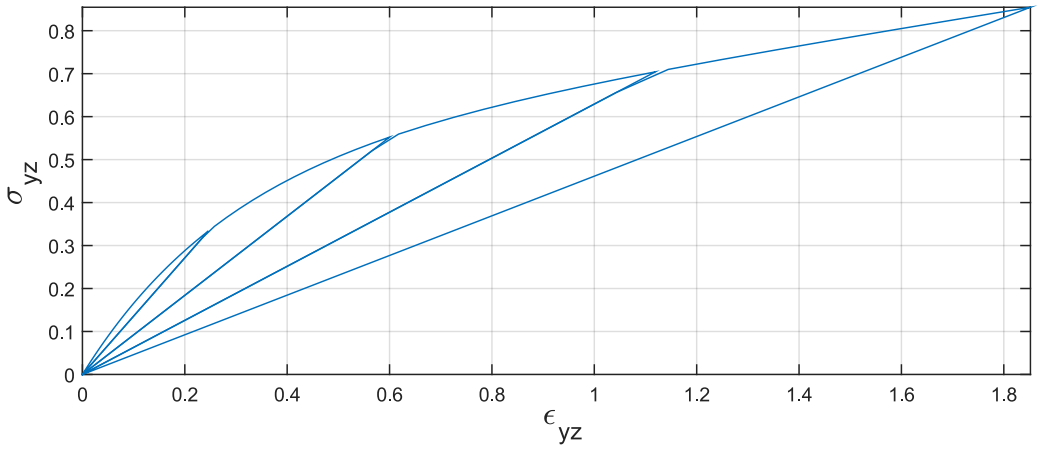

Figure 7. Stress–strain curve.

The constitutive material is linear elastic with Young's modulus $E = 5$ Pa and Poisson's ratio $\nu = 0.3$; and the damage parameters are $\alpha = 0.4$, and $\beta = 0.1$, which are selected so that the model can reach quickly a high level of damage. The POD reduced basis $\boldsymbol{\upsilon}$ is obtained following the strategy described in the previous example, by which a standard FEM model is used as a reference. The chosen POD reduced basis consists of only *one* vector which explains nearly the 100% of the total variance of the 200 provided snapshots. With regard to the coefficient ξ_1, it is uniformly discretised in the interval $[-3, 0]$ with $12\,000$ nodes. In this case, PGD only required 2 modes to reach convergence.

The incorporation of the computational vademecum to Algorithm 1 allows the real-time computation of the 200 pseudo-time step, six of which are shown in Fig. 6. In this example, the damage is uniformly distributed throughout the volume of the model, and intensifies as the magnitude of the load increases. For each simulation step, the tensors of strain $\boldsymbol{\varepsilon}$ and stress $\boldsymbol{\sigma}$ have been computed. The stress–strain curve of the model is plotted in Fig. 7. Lastly, L_2 error norms have been computed for each step (see Fig. 8), which allow to determine how the results obtained by the POD–PGD algorithm differ from the reference solution.

DOI: 10.1002/cnm

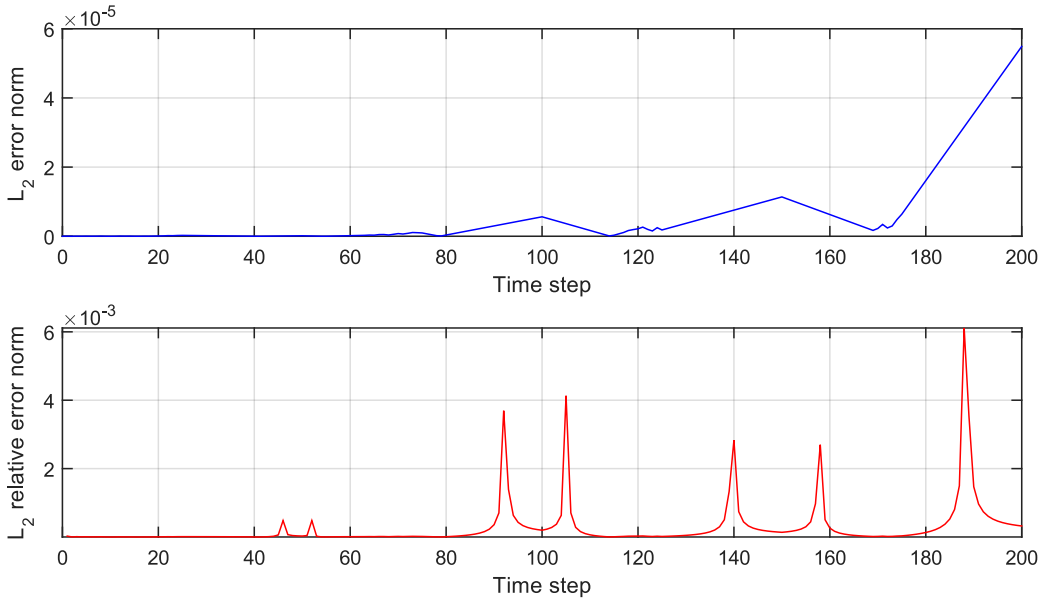

Figure 8. L_2 error norm (top) and L_2 relative error norm (bottom) of the unit cube undergoing pure shear deformation for each of the 200 computed pseudo-time steps.

5.2. Removal of fatty tissue from gallbladder

Some surgical procedures require the removal of the adipose tissue that is adhered to an organ so that it can be accessible. As mentioned before, in laparoscopic cholecystectomy (i. e., the removal of the gallbladder), the laparoscopic instruments can be equipped with either scraping tools or hook-shaped electrocautery tips, which allow to tear the fatty tissue that surrounds the gallbladder (specifically, the region known as the Calot's triangle) prior to its extirpation. In this section, an example of the removal of the fat adjoining the gallbladder by tearing the tissue is considered. This example uses the POD-PGD approach and the simplified model of damage implemented in the previous sections. This simulation has been incorporated into our haptic simulator.

A model of the Calot's triangle region including the common hepatic duct, the cystic duct, and the right hepatic duct, along with the gallbladder is considered [45], see Fig. 9. In white, some adipose tissue has been added to the anatomic model.

The gallbladder is meshed using trilinear hexahedral elements, and consists of 31 128 elements (28 232 for the bladder, 2 896 for the fat) and 37 010 nodes. Two views of the mesh are depicted in Fig. 10. A large part of the nodes on the surface of both the rear part of the gallbladder and its ducts are fully fixed, thus emulating the adherent contact of these areas with the liver and other surrounding connective tissues.

The constitutive material is linear elastic (the consideration of hyperelastic models does not induce any additional difficulty and has been previously studied in [12, 36]). Unlike the previous examples, two different material properties are considered, one for the gallbladder and another for the fat. Experimental studies [46, 47, 48] suggest that the material properties of the gallbladder, although highly variable, can be approximated with values $E_G = 1.15$ kPa and $\nu_G = 0.48$, whereas the fat can be characterized with $E_F = 0.5$ MPa and $\nu_F = 0.495$.

The POD reduced basis of the damage in the fat is obtained following the strategy described in the previous examples, by taking an identical FEM model as a reference. To test the proper performance of the method, traction is applied to one node of the fat tissue region. 15 pseudo-time steps are simulated with constant $\Delta \boldsymbol{f} = 1/3$ N. From the obtained POD solution, a reduced basis consisting of three vectors is selected, which accounts for nearly 100% of the explained variance. Coefficients ξ_i are discretised with the following interval limits, and number of uniformly distributed nodes: $\xi_1 \in [-152.1, -145.2]$ using 6 900 nodes; $\xi_2 \in [-6.3, 5.8]$ using 4 840 nodes; and

DOI: 10.1002/cnm

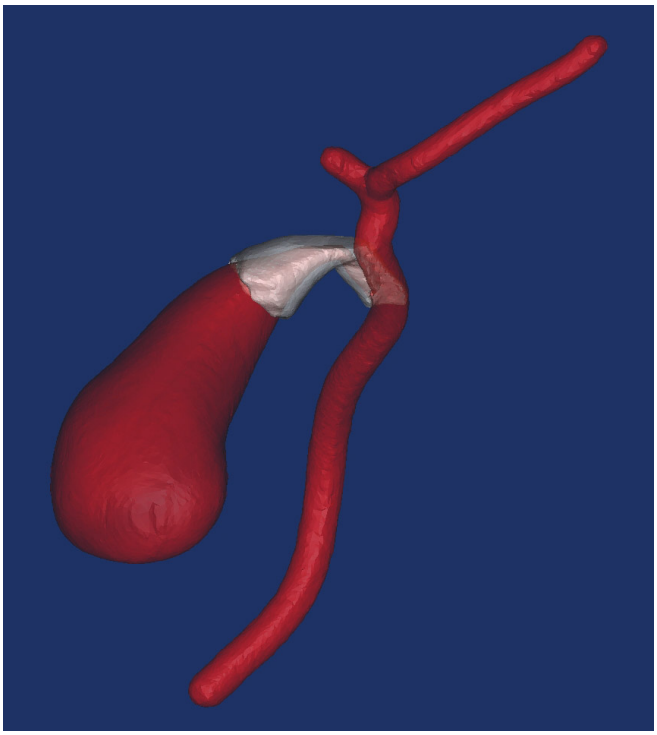

Figure 9. Model of the Calot's triangle region.

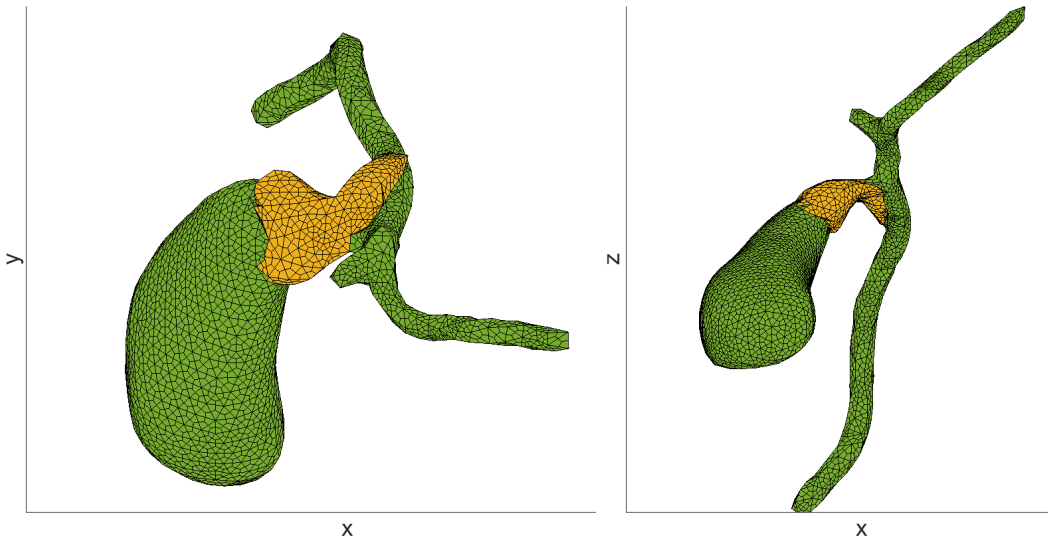

Figure 10. Geometry of the finite element model of the gallbladder (in green) and its adipose tissue (in yellow).

$\xi_3 \in [-0.3, 0.4]$ using 2 800 nodes. With these data, an accurate PGD solution is achieved with 17 modes.

The PGD computational vademecum is then incorporated into the incremental algorithm, and the fifteen pseudo-time steps can be now computed in real-time, under haptic constraints, i.e., in less than one millisecond. The sequence, depicted in Fig. 11, simulates the interactive tearing of the adipose tissue by means of a laparoscopic instrument (not pictured), which grasps and pulls the

DOI: 10.1002/cnm

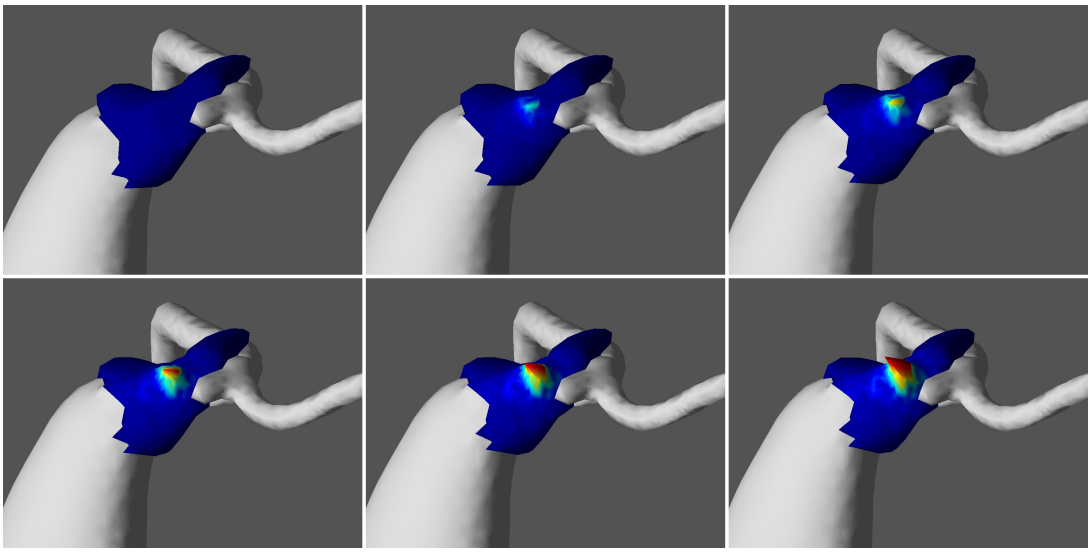

Figure 11. Damage sequence.

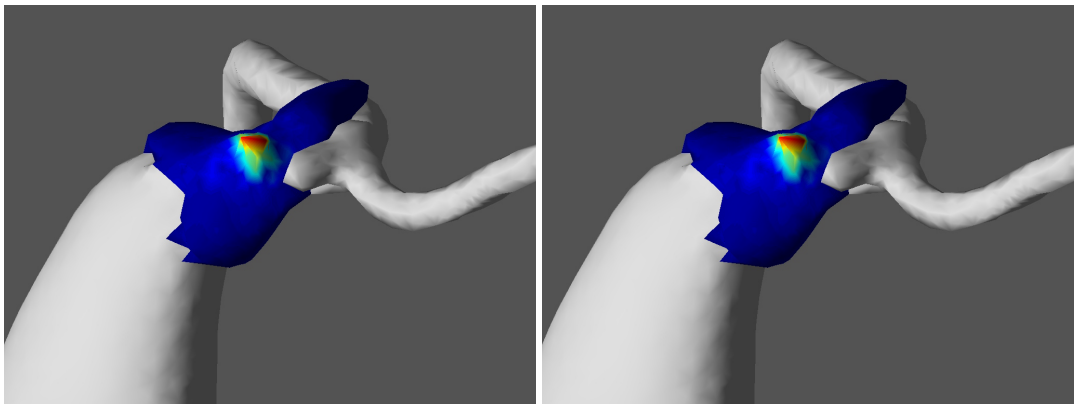

Figure 12. Comparison between the standard FEM (left) and PGD (right) prediction of damage at a given time instant.

tissue outwards. The values of damage in the affected area increase with the total applied load, and levels of damage exceeding certain threshold indicate the full rupture of the tissue.

As a measure of the low error that the POD-PGD approach produces, the tearing sequence computed using this method is visually identical to the one computed with the FEM approach, for both the attained deformations and the damage values. A comparison of damage values for a given time instant is shown in Fig. 12.

A comparison of the difference with the reference, full order, finite element simulation is included in Fig. 13. Similarly, the difference in the predicted displacement, both absolute and relative, is shown in Fig. 14.

For visualization purposes, elements whose Gauss points have reached a prescribed damage threshold, say 0.8, are eliminated from the rendering process. Indeed, grasping is considered lost and therefore elastic recovery is perceived from the user's perspective, see Fig. 15. To increase realism, the damaged surface is rendered with artificial roughness, so as to indicate the user the breaking of the tissue.

DOI: 10.1002/cnm

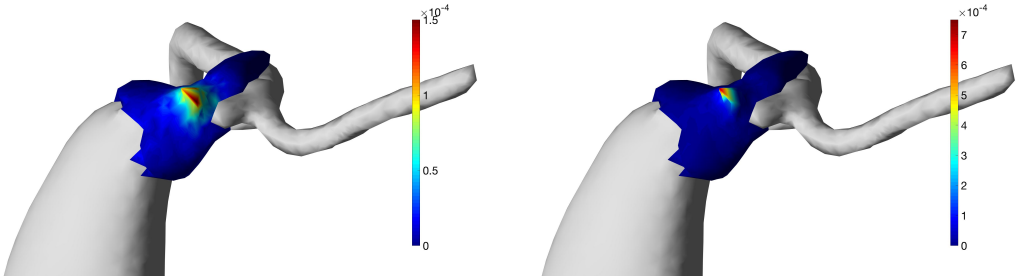

Figure 13. Difference in damage values between the reference FEM simulation and PGD. Absolute difference (left) vs. relative difference (right).

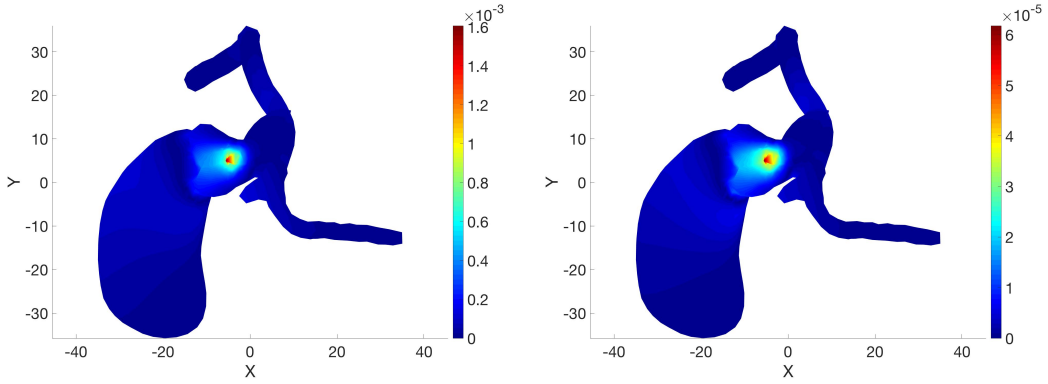

Figure 14. Difference in displacement values between the reference FEM simulation and PGD. Absolute difference (left) vs. relative difference (right).

5.3. Timing

The results presented in this section have been obtained with a 64-bit laptop, running Matlab 2014b with an i5 processor running at 2.50 GHz, 4Gb RAM memory. Despite the use of crude Matlab code prototypes, the examples of the gallbladder ran always faster than 1 kHz, which is enough for visual as well as haptic real-time requirements.

The final version of the simulator, implemented in the OpenHaptics Toolkit [49], also provided results always below the millisecond threshold. The force perceived by the user is always smooth, with no appreciable jumps in the peripheral.

6. CONCLUSIONS

In this work, a method for the real-time simulation of the tearing of soft tissues, based on the continuum damage mechanics theory, is presented. As a first approach to solving the model, explicit incremental finite element algorithms could be considered. However, they require the computation of the inverse of a large stiffness matrix at each step, which makes its implementation incompatible with the real-time requirements of haptic simulators.

To address this problem, the use of PGD computational vademecums is proposed, since they allow to obtain quick direct solutions with a low computational cost. Nevertheless, the parametrization of the damage field for this kind of problems may be huge, and a reduced order model technique is required to minimize the number of parameters of the vademecum. To this end, a reduced basis of the damage field is computed using POD, thus compacting the vademecum to a great extent.

This POD-PGD method is tried in both academic and complex examples with excellent results. A surgical application focused on the simulation of the tearing of adipose tissue during cholecystectomy is implemented.

DOI: 10.1002/cnm

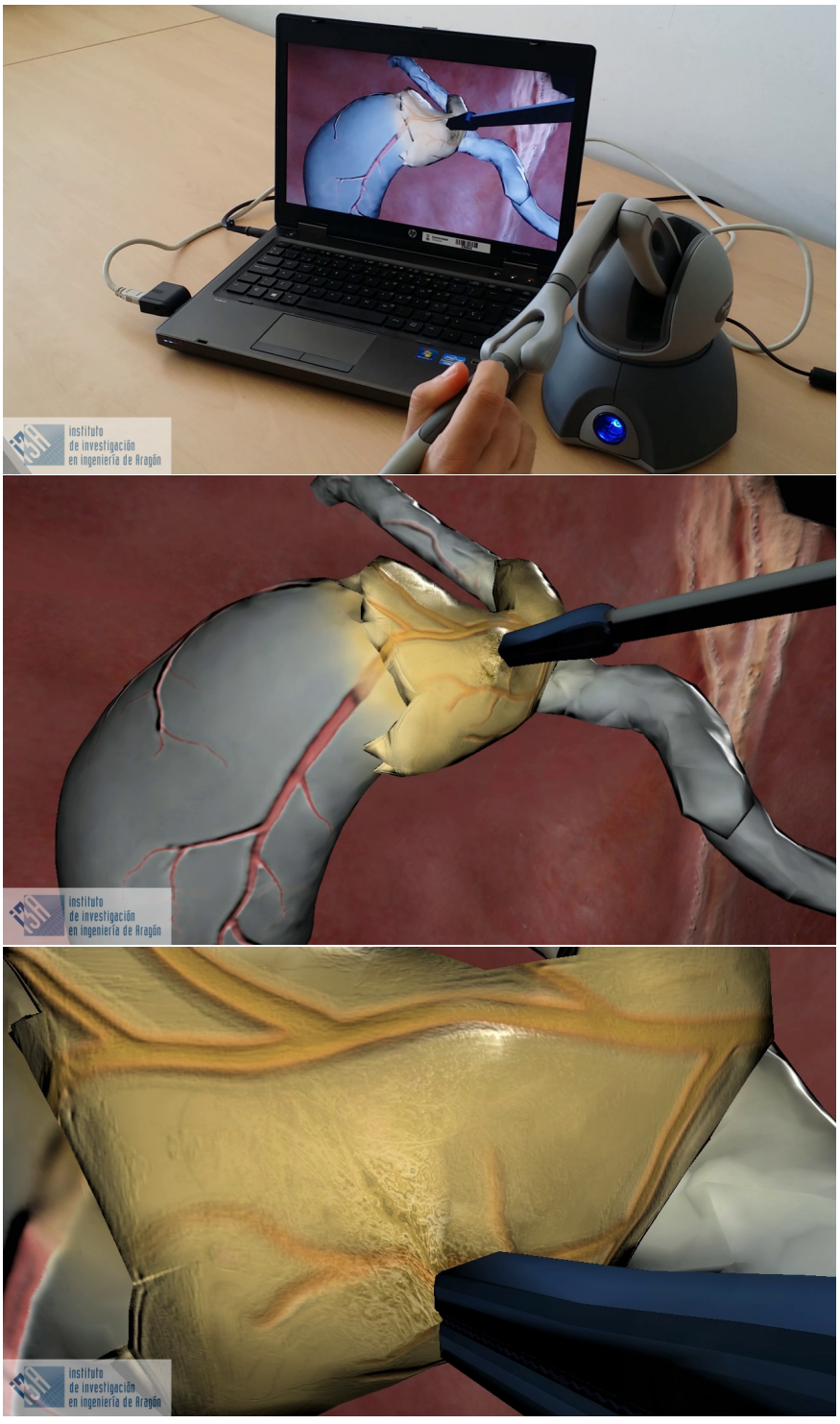

Figure 15. Some snapshots of the implemented simulator.

The reduced computing times achieved by the implemented algorithm allow a real-time performance compatible with haptic environments, and the levels of error presented by the method are low when compared with reference FEM models.

DOI: 10.1002/cnm

The proposed method renders a high degree of realism in the simulation of tearing of soft tissue. In sharp contrast to existing approaches, our proposed method does not need for any update in the mesh of the model. No element is modified or deleted. The method, although initially explicit, could be extended to purely implicit frameworks with no extra cost. This is possible since computational vademecums are solved entirely off-line, and their results particularized in the on-line phase of the execution pipeline. These improvements constitute currently our main topic of research and their results will be published elsewhere.

A. MATRIX FORM OF THE POD-PGD METHOD

In what follows we include the matrix form of the method just developed for the fixed point iteration algorithm. For the sake of readability, we consider the simplified case in which $h = 0$. Also for clarity of exposition in this formulation, parameter $\boldsymbol{s}$ governing the position of the load is also omitted. A detailed treatment of the force value and position as parameter within PGD can be found in [30].

Assuming that the POD basis of the scalar field of damage $\boldsymbol{g}$ consists of q vectors, Eq. (15 left), and by discarding all the integrals and variables related to the scalar field of damage $\boldsymbol{h}$, the weak form of the simplified model can be obtained from Eq. (16) as

$$\int_{\mathcal{E}_q} \cdots \int_{\mathcal{E}_1} \int_{\Omega} \mathcal{U} \; \mathrm{d}\Omega \, \mathrm{d}\mathcal{E}_1 \cdots \mathrm{d}\mathcal{E}_q = \int_{\mathcal{E}_q} \cdots \int_{\mathcal{E}_1} \int_{\Gamma_{t_2}} \Delta \boldsymbol{u}^* \cdot \Delta \boldsymbol{f} \, \mathrm{d}\Gamma \, \mathrm{d}\mathcal{E}_1 \cdots \mathrm{d}\mathcal{E}_q, \tag{19}$$

with $\mathcal{U} = \boldsymbol{\nabla}_{\mathrm{s}} \, \Delta \boldsymbol{u}^* : \boldsymbol{g} \, \mathbf{C}_0 : \boldsymbol{\nabla}_{\mathrm{s}} \, \Delta \boldsymbol{u}$.

The discrete form for the computation of a new mode of $\Delta \boldsymbol{u}$ in Eq. (18) can be expressed, following the notation already described [30], as

$$\Delta \boldsymbol{u}(\boldsymbol{x}; \xi_1, \ldots, \xi_q) = \sum_{i=1}^{n} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \boldsymbol{F}^i \circ \bigodot_{K=1}^{q} \boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K) \, \boldsymbol{G}_K{}^i + \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \boldsymbol{R} \circ \bigodot_{K=1}^{q} \boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K) \, \boldsymbol{S}_K, \tag{20}$$

where "$\circ$" represents the entry-wise Hadamard or Schur multiplication of vectors, "$\bigodot$" is its associated product symbol, $\boldsymbol{M}_K$ represents the vectors $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_q$, which contain the one-dimensional finite element shape functions of the parameters $\xi_1, \xi_2, \ldots, \xi_q$, respectively; and $\boldsymbol{G}_K{}^i$ represents the vectors $\boldsymbol{G}_1{}^i, \boldsymbol{G}_2{}^i, \ldots, \boldsymbol{G}_q{}^i$, which contain the nodal values of the finite element mesh of the functions $\boldsymbol{G}_1{}^i(\xi_1), \boldsymbol{G}_2{}^i(\xi_2), \ldots, \boldsymbol{G}_q{}^i(\xi_q)$, respectively. The latter also applies in the case of $\boldsymbol{S}_K$.

The admissible variation of $\Delta \boldsymbol{u}$, in turn, can be expressed as

$$\begin{aligned} \Delta \boldsymbol{u}^*(\boldsymbol{x}; \xi_1, \ldots, \xi_q) = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \boldsymbol{R}^* \circ \bigodot_{K=1}^{q} \boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K) \, \boldsymbol{S}_K \\ + \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \boldsymbol{R} \circ \boldsymbol{M}_1{}^{\mathrm{T}}(\xi_1) \, \boldsymbol{S}_1{}^* \circ \ldots \circ \boldsymbol{M}_q{}^{\mathrm{T}}(\xi_q) \, \boldsymbol{S}_q + \ldots \\ + \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \boldsymbol{R} \circ \boldsymbol{M}_1{}^{\mathrm{T}}(\xi_1) \, \boldsymbol{S}_1 \circ \ldots \circ \boldsymbol{M}_q{}^{\mathrm{T}}(\xi_q) \, \boldsymbol{S}_q{}^*. \end{aligned} \tag{21}$$

The fixed-point iteration implies the computation of $\boldsymbol{R}, \boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_q$ when, for each of the variables, the remaining ones are known. This computation is described below. In the case of the variables $\boldsymbol{S}_K$, with $K = 1, 2, \ldots, a, \ldots, q$, the development of the expression for an arbitrary $\boldsymbol{S}_a$ is provided.

A.1. Computation of $\boldsymbol{R}$ when the rest of the variables are known

If $\boldsymbol{R}$ is searched when $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_q$ are known, then $\boldsymbol{S}_1{}^* = \boldsymbol{S}_2{}^* = \ldots = \boldsymbol{S}_q{}^* = \mathbf{0}$ and Eq. (21) can be expressed as

$$\Delta \boldsymbol{u}^*(\boldsymbol{x}; \xi_1, \ldots, \xi_q) = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x}) \, \boldsymbol{R}^* \circ \bigodot_{K=1}^{q} \boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K) \, \boldsymbol{S}_K. \tag{22}$$

Prepared using cnmauth.cls
DOI: 10.1002/cnm

By incorporating Eqs. (15 left), (20), and (22) into the left-hand side of Eq. (19), $\mathcal{U}$ results in

$$\mathcal{U} = \boldsymbol{\nabla}_{\mathrm{s}} \left(\boldsymbol{R}^{*\mathrm{T}}\, \boldsymbol{N}(\boldsymbol{x}) \circ \bigodot_{K=1}^{q} \boldsymbol{S}_K{}^{\mathrm{T}}\, \boldsymbol{M}_K(\xi_K) \right) \cdot \left(\mathbf{C}_0 \cdot \sum_{i=1}^{q} \xi_i \cdot \boldsymbol{v}_i \right) \cdot \boldsymbol{\nabla}_{\mathrm{s}} \left(\sum_{i=1}^{n} \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\, \boldsymbol{F}^i \circ \bigodot_{K=1}^{q} \boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K)\, \boldsymbol{G}_K{}^i + \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\, \boldsymbol{R} \circ \bigodot_{K=1}^{q} \boldsymbol{M}_K{}^{\mathrm{T}}(\xi_K)\, \boldsymbol{S}_K \right);$$

and the right-hand side of Eq. (19), after substituting Δu^* by Eq. (22), can be expressed as

$$\int_{\mathcal{E}_q} \cdots \int_{\mathcal{E}_1} \int_{\Gamma_{t_2}} \left(\boldsymbol{R}^{*\mathrm{T}}\, \boldsymbol{N}(\boldsymbol{x}) \circ \bigodot_{K=1}^{q} \boldsymbol{S}_K{}^{\mathrm{T}}\, \boldsymbol{M}_K(\xi_K) \right) \cdot \Delta \boldsymbol{f}\ \mathrm{d}\Gamma\, \mathrm{d}\mathcal{E}_1 \cdots \mathrm{d}\mathcal{E}_q.$$

Finally, after applying variable separation, rearranging the terms, and calculating the integrals, Eq. (19) can be written as

$$\begin{aligned} \sum_{i=1}^{n} \left(\boldsymbol{R}^{*\mathrm{T}}\, \boldsymbol{K_1}\, \boldsymbol{F}^i \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi_1}}\, \boldsymbol{G}_1{}^i \right) \cdots \left(\boldsymbol{S}_q{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}q}\, \boldsymbol{G}_q{}^i \right) + \ldots \\ + \sum_{i=1}^{n} \left(\boldsymbol{R}^{*\mathrm{T}}\, \boldsymbol{K}_q\, \boldsymbol{F}^i \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}1}\, \boldsymbol{G}_1{}^i \right) \cdots \left(\boldsymbol{S}_q{}^{\mathrm{T}} \boldsymbol{M}_{\boldsymbol{\xi}_q}\, \boldsymbol{G}_q{}^i \right) \\ + \left(\boldsymbol{R}^{*\mathrm{T}}\, \boldsymbol{K}_1\, \boldsymbol{R} \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi}_1}\, \boldsymbol{S}_1 \right) \cdots \left(\boldsymbol{S}_q{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}q}\, \boldsymbol{S}_q \right) + \ldots \\ + \left(\boldsymbol{R}^{*\mathrm{T}}\, \boldsymbol{K}_q\, \boldsymbol{R} \right) \cdot \left(\boldsymbol{S}_1{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}1}\, \boldsymbol{S}_1 \right) \cdots \left(\boldsymbol{S}_q{}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi}_q}\, \boldsymbol{S}_q \right) \\ = \boldsymbol{R}^{*\mathrm{T}}\, \Delta \boldsymbol{f} \cdot \boldsymbol{S}_1{}^{\mathrm{T}}\, \boldsymbol{m}_{\boldsymbol{G}1} \cdot \ldots \cdot \boldsymbol{S}_q{}^{\mathrm{T}}\, \boldsymbol{m}_{\boldsymbol{G}q}, \end{aligned} \quad (23)$$

where the matrices $\boldsymbol{K}_K$, with $K = 1, 2, \ldots, q$, represent the integrals

$$\boldsymbol{K}_K = \int_{\Omega} \boldsymbol{v}_K\, \boldsymbol{B}^{\mathrm{T}}\, \mathbf{C} \boldsymbol{B}\, \mathrm{d}\Omega, \quad (24)$$

with $\boldsymbol{B}$ being the shape function derivative matrix, and $\boldsymbol{v}_K$ being the K-th vector of the reduced-order basis, see Eq. (15 left). The matrices $\boldsymbol{M}_{\boldsymbol{G}L}$, with $L = 1, 2, \ldots, q$, represent the mass matrices of the one-dimensional discretisations of the parameters ξ_L. For each term in Eq. (23), the matrix $\boldsymbol{M}_{\boldsymbol{G}L}$ does not exist when $L = K$. Instead, matrices $\boldsymbol{M}_{\boldsymbol{\xi}\boldsymbol{K}}$ are present, which have the form

$$\boldsymbol{M}_{\boldsymbol{\xi}K} = \int_{\mathcal{E}_K} \xi_K\, \boldsymbol{M}_K\, \boldsymbol{M}_K{}^{\mathrm{T}}\, \mathrm{d}\mathcal{E}_K.$$

Lastly, $\boldsymbol{m}_{\boldsymbol{G}K}$ are vectors containing the values of the finite element shape functions, $\int_{\mathcal{E}_K} \boldsymbol{M}_K\, \mathrm{d}\mathcal{E}_K$. Given these data, an expression of $\boldsymbol{R}$ can be obtained from Eq. (23).

A.2. Computation of an arbitrary $\boldsymbol{S_a}$ when the rest of the variables are known

Similarly, if an arbitrary $\boldsymbol{S_a}$ is searched when the rest of the variables are known, then $\boldsymbol{R}^* = \boldsymbol{S}_1{}^* = \ldots = \boldsymbol{S}_{a-1}{}^* = \boldsymbol{S}_{a+1}{}^* = \ldots = \boldsymbol{S}_q{}^* = \mathbf{0}$, and Eq. (21) can be expressed as

$$\Delta \boldsymbol{u}^*(\boldsymbol{x}; \xi_1, \ldots, \xi_q) = \boldsymbol{N}^{\mathrm{T}}(\boldsymbol{x})\, \boldsymbol{R} \circ \boldsymbol{M}_1{}^{\mathrm{T}}(\xi_1)\, \boldsymbol{S}_1 \circ \ldots \circ \boldsymbol{M}_a{}^{\mathrm{T}}(\xi_a)\, \boldsymbol{S}_a{}^* \circ \ldots \circ \boldsymbol{M}_q{}^{\mathrm{T}}(\xi_q)\, \boldsymbol{S}_q. \quad (25)$$

DOI: 10.1002/cnm

By incorporating Eqs. (15 left), (20), and (25) into Eq. (19), and operating as in the previous section, the resulting equation is

$$\sum_{i=1}^{n} \left(\boldsymbol{R}^{\mathrm{T}}\, \boldsymbol{K}_1\, \boldsymbol{F}^i\right) \cdot \left({\boldsymbol{S}_1}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi}_1}\, {\boldsymbol{G}_1}^i\right) \cdots \left({\boldsymbol{S}_a}^{*\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}a}\, {\boldsymbol{G}_a}^i\right) \cdots$$
$$\cdots \left({\boldsymbol{S}_{\boldsymbol{K}}}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}q}\, {\boldsymbol{G}_q}^i\right) + \ldots + \sum_{i=1}^{n} \left(\boldsymbol{R}^{\mathrm{T}}\, \boldsymbol{K}_q\, \boldsymbol{F}^i\right) \cdot \left({\boldsymbol{S}_1}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}1}\, {\boldsymbol{G}_1}^i\right) \cdots$$
$$\cdots \left({\boldsymbol{S}_a}^{*\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}a}\, {\boldsymbol{G}_a}^i\right) \cdots \left({\boldsymbol{S}_q}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi}_q}\, {\boldsymbol{G}_q}^i\right) + \left(\boldsymbol{R}^{\mathrm{T}}\, \boldsymbol{K}_1\, \boldsymbol{R}\right) \cdot$$
$$\cdot \left({\boldsymbol{S}_1}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi}_1}\, \boldsymbol{S}_1\right) \cdots \left({\boldsymbol{S}_a}^{*\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}a}\, \boldsymbol{S}_a\right) \cdots \left({\boldsymbol{S}_q}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}q}\, \boldsymbol{S}_q\right) + \ldots +$$
$$+ \left(\boldsymbol{R}^{\mathrm{T}}\, \boldsymbol{K}_q\, \boldsymbol{R}\right) \cdot \left({\boldsymbol{S}_1}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}1}\, \boldsymbol{S}_1\right) \cdots \left({\boldsymbol{S}_a}^{*\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{G}a}\, \boldsymbol{S}_a\right) \cdots$$
$$\cdots \left({\boldsymbol{S}_q}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{\xi}_q}\, \boldsymbol{S}_q\right) = \boldsymbol{R}^{\mathrm{T}}\, \Delta \boldsymbol{f} \cdot {\boldsymbol{S}_1}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{S}} \cdot \ldots \cdot {\boldsymbol{S}_a}^{\mathrm{T}}\, \boldsymbol{M}_a \cdot \ldots \cdot {\boldsymbol{S}_{\boldsymbol{K}}}^{\mathrm{T}}\, \boldsymbol{M}_{\boldsymbol{S}},$$

from which an expression for $\boldsymbol{S}_a$ can be found.

REFERENCES

1. Stephane Cotin, Herve Delingette, and Nicholas Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999.
2. Hervé Delingette and Nicholas Ayache. Soft tissue modeling for surgery simulation. *Handbook of Numerical Analysis*, 2004.
3. Guillaume Picinbono, Hervé Delingette, and Nicholas Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *Graphical models*, 65(5):305–321, 2003.
4. Suvranu De, Yi-Je Lim, Muniyandi Manivannan, and Mandayam a Srinivasan. Physically Realistic Virtual Surgery Using the Point-Associated Finite Field (PAFF) Approach. *Presence: Teleoperators and Virtual Environments*, 15(3):294–308, 2006.
5. K. Miller, G. Joldes, D. Lance, and A. Wittek. Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Commun. Numer. Methods Eng.*, 23(2):121–134, 2007.
6. S. Niroomandi, I. Alfaro, E. Cueto, and F. Chinesta. Real-time deformable models of non-linear tissues by model reduction techniques. *Computer Methods and Programs in Biomedicine*, 91(3):223–231, 2008.
7. Elías Cueto and Francisco Chinesta. Real time simulation for computational surgery: a review. *Advanced Modeling and Simulation in Engineering Sciences*, 1(1):11, 2014.
8. Hadrien Courtecuisse, Jérémie Allard, Pierre Kerfriden, Stéphane P.A. Bordas, Stéphane Cotin, and Christian Duriez. Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical Image Analysis*, 18(2):394 – 410, 2014.
9. Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Computer graphics forum*, volume 15, pages 57–66. Wiley Online Library, 1996.
10. Jérémie Allard, Stéphane Cotin, François Faure, Pierre-Jean Bensoussan, François Poyer, Christian Duriez, Hervé Delingette, and Laurent Grisoni. Sofa-an open source framework for medical simulation. In *MMVR 15-Medicine Meets Virtual Reality*, volume 125, pages 13–18. IOP Press, 2007.
11. Olivier Comas, Zeike A Taylor, Jérémie Allard, Sébastien Ourselin, Stéphane Cotin, and Josh Passenger. Efficient nonlinear fem for soft tissue modelling and its gpu implementation within the open source framework sofa. In *Biomedical Simulation*, pages 28–39. Springer, 2008.
12. Siamak Niroomandi, Icíar Alfaro, David González, Elías Cueto, and Francisco Chinesta. Model order reduction in hyperelasticity: a proper generalized decomposition approach. *International Journal for Numerical Methods in Engineering*, 96(3):129–149, 2013.
13. David González, Icíar Alfaro, Carlos Quesada, Elías Cueto, and Francisco Chinesta. Computational vademecums for the real-time simulation of haptic collision between nonlinear solids. *Computer Methods in Applied Mechanics and Engineering*, 283:210–223, 2015.
14. Suleiman M. BaniHani and Suvranu De. A comparison of some model order reduction methods for fast simulation of soft tissue response using the point collocation-based method of finite spheres. *Engineering with Computers*, 25(1):37–47, 2009.
15. David González, Elías Cueto, and Francisco Chinesta. Real-time direct integration of reduced solid dynamics equations. *International Journal for Numerical Methods in Engineering*, 99(9):633–653, 2014.
16. Daniel Bielser, Volker A Maiwald, and Markus H Gross. Interactive cuts through 3-dimensional soft tissue. In *Computer Graphics Forum*, volume 18, pages 31–38. Wiley Online Library, 1999.
17. Cynthia D Bruyns, Steven Senger, Anil Menon, Kevin Montgomery, Simon Wildermuth, and Richard Boyle. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The journal of visualization and computer animation*, 13(1):21–42, 2002.

DOI: 10.1002/cnm

18. Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.
19. Hadrien Courtecuisse, Jérémie Allard, Pierre Kerfriden, Stéphane PA Bordas, Stéphane Cotin, and Christian Duriez. Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical image analysis*, 18(2):394–410, 2014.
20. Jun Wu, Rüdiger Westermann, and Christian Dick. Physically-based simulation of cuts in deformable bodies: A survey. In *Eurographics (State of the Art Reports)*, pages 1–19. Citeseer, 2014.
21. Lenka Jeřábková and Torsten Kuhlen. Stable cutting of deformable objects in virtual environments using xfem. *IEEE Computer Graphics and Applications*, 2:61–71, 2009.
22. S. Niroomandi, I. Alfaro, D. González, E. Cueto, and 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):574–588, 2012.
23. Jérémie Allard, Maud Marchal, and Stéphane Cotin. Fiber-based Fracture Model for Simulating Soft Tissue Tearing. *Studies in Health Technology and Informatics*, 142:13–18, 2009.
24. Kumar Vemaganti, Grand Joldes, Karol Miller, and Adam Wittek. *Total Lagrangian Explicit Dynamics-Based Simulation of Tissue Tearing*, volume Part 1, pages 63–72. Springer Science+Business Media, 2011.
25. Jakub Kaczynski and Joanna Hilton. A gallbladder with the "hidden cystic duct": A brief overview of various surgical techniques of the calot's triangle dissection. *Interventional Medicine and Applied Science*, 7(1):42–45, mar 2015.
26. Andrés Mena, David Bel, Icíar Alfaro, David González, Elías Cueto, and Francisco Chinesta. Towards a pancreatic surgery simulator based on model order reduction. *Advanced Modeling and Simulation in Engineering Sciences*, 2(1):31, 2015.
27. F. Chinesta, A. Leygue, F. Bordeu, J. V. Aguado, E. Cueto, D. Gonzalez, I. Alfaro, A. Ammar, and A. Huerta. PGD-Based Computational Vademecum for Efficient Design, Optimization and Control. *Archives of Computational Methods in Engineering*, 20(1):31–59, 2013.
28. F. Chinesta, R. Keunings, and A. Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer*. SpringerBriefs in Applied Sciences and Technology. Springer International Publishing, 2013.
29. Francisco Chinesta and Elías Cueto. *PGD-based modeling of materials, structures and processes*. Springer, 2014.
30. E. Cueto, D. González, and I. Alfaro. *Proper Generalized Decompositions: An Introduction to Computer Implementation with Matlab*. SpringerBriefs in Applied Sciences and Technology. Springer International Publishing, 2016.
31. J. Lemaitre. *A Course on Damage Mechanics*. Springer Berlin Heidelberg, 1996.
32. Juan Carlos Simó. On a fully three-dimensional finite-strain viscoelastic damage model: formulation and computational aspects. *Computer methods in applied mechanics and engineering*, 60(2):153–173, 1987.
33. A. Ammar, F. Chinesta, P. Diez, and A. Huerta. An error estimator for separated representations of highly multidimensional models. *Computer Methods in Applied Mechanics and Engineering*, 199(25-28):1872 – 1880, 2010.
34. Icíar Alfaro, David González, Sergio Zlotnik, Pedro Díez, Elías Cueto, and Francisco Chinesta. An error estimator for real-time simulators based on model order reduction. *Advanced Modeling and Simulation in Engineering Sciences*, 2(1):30, 2015.
35. Amine Ammar, B Mokdad, Francisco Chinesta, and Roland Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids: Part II: Transient simulation using space-time separated representations. *Journal of Non-Newtonian Fluid Mechanics*, 144(2):98–121, 2007.
36. S Niroomandi, D González, I Alfaro, F Bordeu, Adrien Leygue, E Cueto, and Francisco Chinesta. Real-time simulation of biological soft tissues: a PGD approach. *International journal for numerical methods in biomedical engineering*, 29(5):586–600, 2013.
37. Bruno Cochelin, Noureddine Damil, and Michel Potier-Ferry. The asymptotic-numerical method: an efficient perturbation technique for nonlinear structural mechanics. *Revue européenne des éléments finis*, 3(2):281–297, 1994.
38. José V Aguado, Francisco Chinesta, Adrien Leygue, Elias Cueto Prendes, and Antonio Huerta. Deim-based pgd for parametric nonlinear model order reduction. In *Adaptive Modeling and Simulation 2013*, pages 29–34. Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), 2013.
39. Lenka Jeřábková, Guillaume Bousquet, Sébastien Barbier, François Faure, and Jérémie Allard. Volumetric modeling and interactive cutting of deformable bodies. *Progress in biophysics and molecular biology*, 103(2):217–224, 2010.
40. Carlos Quesada, David González, Iciar Alfaro, Elías Cueto, and Francisco Chinesta. Computational vademecums for real-time simulation of surgical cutting in haptic environments. *International Journal for Numerical Methods in Engineering*, pages n/a–n/a, 2016. nme.5252.
41. G.Z. Voyiadjis and P.I. Kattan. *Damage Mechanics*. Mechanical Engineering. CRC Press, 2005.
42. J Lemaitre and JL Chaboche. *Mécanique des matériaux solides*. Dunod, Paris, 1985.
43. Jérémie Allard, Maud Marchal, Stéphane Cotin, et al. Fiber-based fracture model for simulating soft tissue tearing. *Studies in health technology and informatics*, 142:13–18, 2009.
44. Alberto Badías, David González, Iciar Alfaro, Francisco Chinesta, and Elias Cueto Prendes. Local proper generalized decomposition. *International Journal for Numerical Methods in Engineering*, pages n/a–n/a. nme.5578.
45. The Database Center for Life Science Japan. Bodyparts3d. Licensed under CC Attribution-Share Alike 2.1, 2015.
46. Nadia Alkhouli, Jessica Mansfield, Ellen Green, James Bell, Beatrice Knight, Neil Liversedge, Ji Chung Tham, Richard Welbourn, Angela C Shore, Katarina Kos, et al. The mechanical properties of human adipose tissues and

DOI: 10.1002/cnm

their relationships to the structure and composition of the extracellular matrix. *American Journal of Physiology-Endocrinology and Metabolism*, 305(12):E1427–E1435, 2013.
47. Kerstyn Comley and Norman A Fleck. A micromechanical model for the young's modulus of adipose tissue. *International Journal of Solids and Structures*, 47(21):2982–2990, 2010.
48. Amit Gefen and Einat Haberman. Viscoelastic properties of ovine adipose tissue covering the gluteus muscles. *Journal of biomechanical engineering*, 129(6):924–930, 2007.
49. Geomagic. *OpenHaptics Toolkit*. 3D systems - Geomagic solutions, 430 Davis Drive, Suite 300 Morrisville, NC 27560 USA, 2013.

DOI: 10.1002/cnm